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

M M Mahabubur Rahman[1*], Debojyoti Biswas[1], Xiao Chen[1], Jelena Tešić[1]

[1]Computer Science, Texas State University, 601 University Dr, San Marcos, 78666, Texas, USA.

*Corresponding author(s). E-mail(s): toufik@txstate.edu;
Contributing authors: debojyoti_biswas@txstate.edu; xc10@txstate.edu;
jtesic@txstate.edu;

Abstract

Deep Neural Networks (DNNs) have gained widespread popularity for tasks related to visual processing due to their superior performance and the wealth of images and videos available. Rich concept representation in the training dataset is crucial for the effectiveness of the trained deep neural network model. Often, images of the same object taken from slightly different angles or with minor variations are present, and this redundancy is wasteful as the bandwidth, storage, and processing power are limited. Near duplicate images contribute very little to the effectiveness of the model, and here we propose a novel framework for Visual Indexing and Retrieval-based image Deduplication (VIRD). VIRD effectively eliminates redundant data and maintains information quality in the training corpus through visual indexing and retrieval. VIRD balances the tradeoff between a large deduplication ratio and a stable mAP by adjusting the deduplication threshold for graph-based approximate retrieval of near-duplicate images from given target corpora. The effectiveness of VIRD is validated through extensive experiments on well-known Convolutional Neural Network (CNN) benchmarks. While preserving the same validation mean Average Precision (mAP), VIRD can reduce the corpus size by 25.13%. Moreover, by streamlining the training process, VIRD can lower the energy consumption of DNN training by 27.17%, leading to more practical and sustainable DNN training practices.

Keywords: Graph-based indexing, Data redundancy, Approximate search, Image deduplication, Near-duplicate images.

1 Introduction

Deep neural networks (DNNs) have significantly impacted the solutions to natural language processing, computer vision, and reinforcement learning tasks by leveraging vast amounts of data. DNNs have demonstrated near-human performance across various vision tasks with tens of millions of images available in the training phase, e.g., ImageNet[1] benchmark. Going beyond known curated image benchmarks, the quality and the percentage of redundancy in a typical dataset are unknown [2]. The SVHN [3] corpus has cropped digits from the house number plates captured by the street view cameras. 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 [2].

Definition 1: *Duplicate or near-duplicate images.* An image I', is considered a duplicate or near duplicate of an image I, if I' is obtained from I through a set of transformations T_r: $I' = T_r(I)$.

Redundant images can result in longer training times of the network, increased storage and memory requirements, and higher energy consumption [4]. We can achieve the same efficiency as the DNN model if it is trained on the smaller yet diversified dataset [5]. Thus, we propose to develop an efficient method capable of identifying and removing redundant data while preserving the effectiveness of the model.

Image features are compact descriptive representations of the objects and scenes in images as they capture 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[6, 7]. In general, the similarity of any object, scene, or global 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.* $D(X, Y)$ is the distance between feature vectors X and Y. The x_i is i^{th} component in feature vector X:

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

When $p = 2$, $D_2(X, Y)$ is known as Euclidean distance. The Euclidean distance-based similarity search is used as a baseline for deep features [8]. Next, we propose the following deduplication rule definition:

Definition 3: *Deduplication rule.* Images X and Y that satisfy

$$D_2(X, Y) \leq T$$

are near duplicate images. T is the threshold for determining image similarity and can be manually adjusted.

In this paper, we propose a **V**isual **I**ndexing and **R**etrieval-based image **D**eduplication, method or the **VIRD** method in short. The VIRD method efficiently discovers and removes near-duplicates from the training image databases while preserving the data integrity. Figure 1 illustrates the dynamic deduplication process. VIRD uses a robust DNN model 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 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. 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 [9–11]. The effectiveness of HLG depends on the DNN model used to extract the deep features. VIRD introduces a novel method by integrating ANNS into a deep feature database, which has a minimal impact on the pre-trained network and training data. ANNS proves highly effective in identifying previously unknown classes within the database [12]. 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.

Paper contributions 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 ample 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.

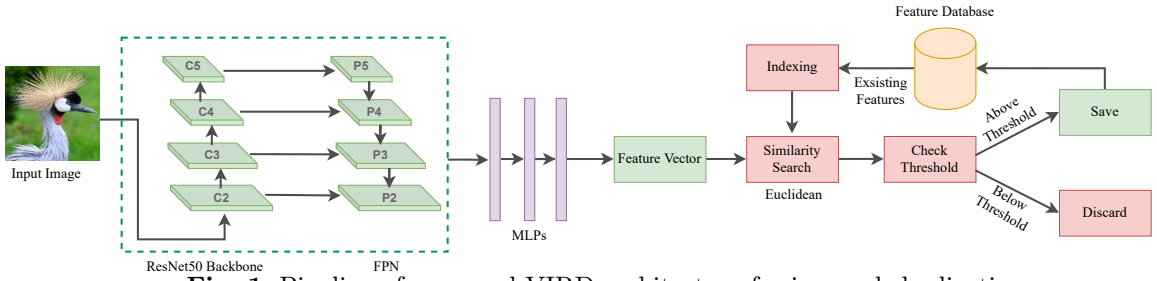

Fig. 1: Pipeline of proposed VIRD architecture for image deduplication.

Section 2 summarizes related work, and section 3 discusses our proposed VIRD method and training pipeline in detail. Next, Section 4 describes the experimental corpus distribution and characteristics. In Section 5, we evaluate the proposed framework and show our experimental results by comparing it with the latest deduplication benchmarks over four consumer and crowd-sensing corpora. Finally, we summarize the findings in Section 6.

2 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. Velmurugan et al. [13] utilized the Speeded-up Robust Features (SURF) algorithm and a k-dimensional (KD) tree for indexing and matching similar image features. Lei et al. [14] introduced a cluster of uniform randomized trees for rapid near-duplicate image detection. Yu et al. [15] presented a SIFT-based geometric transformation fingerprinting technique, while Li et al. [16] proposed an image-matching algorithm using SURF feature points and daisy descriptors. Foo et al. [17] developed a similar image collator (SICO) utilizing PCA-SIFT and LSH for feature indexing. Chen et al. [18] introduced a rapid image retrieval method converting features into binary representations. Deep feature-based methods leverage Convolutional Neural Networks (CNNs) for image deduplication. For instance, Kaur et al. [2] proposed a CNN-based online image deduplication technique, showing superior performance in detecting exact and near-exact images. Kordopatis-Zilos et al. [19] introduced a method for near-duplicate video retrieval using unsupervised and supervised approaches based on Deep Metric Learning (DML). Liang et al. [20] presented a hierarchical detection method utilizing CNN models and semantic features for near-duplicate video identification.

3 Methodology

Exact feature matching is slow when dealing with high-dimensional features and large databases [21–23].

First, the VIRD algorithm 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. This trained DNN acts like a powerful feature extractor, capturing inherent similarities during the feature extraction process. It is important to note that this model does not detect specific objects or features; it focuses on learning general similarities within the data, as shown in section 5. Second, VIRD retrieves the most similar feature using a Hierarchical Layered Graph (HLG) search, followed by the exact similarity measure of retrieved features to the incoming data deep feature. 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. Thus, VIRD effectively reduces storage requirements and processing overhead associated with redundant data. By eliminating duplicate or near-duplicate images, the corpus becomes more compact and efficient, improving visual applications' overall performance.

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. The second phase employs an approximate search in deep descriptor space searching method to find the most similar image features concerning the query

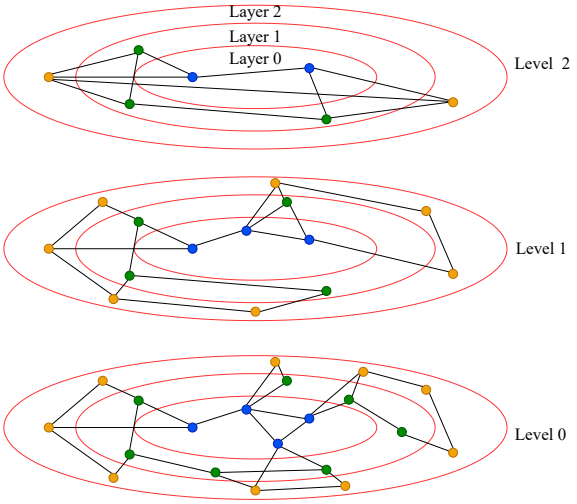

Fig. 2: Hierarchical Layered Graph (HLG) indexing: each feature vector connects to its M-Nearest Neighbor within the same layer and 1-Nearest Neighbor in the next layers.

feature. During the approximate search using HLG, the most similar image feature is retrieved from the feature database for an incoming query image feature.

3.1 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 [24, 25] for object detection in consumer and aerial images, we have chosen to use ResNet50 [26] 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 initial seven blocks within the ResNet50 network are Convolutional (Conv) blocks with 64 approximate searches in profound descriptor space 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, it performs an 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.

3.2 Hierarchical Layered Graph (HLG)

HLG index building Figure 2 depicts the index structure of our HLG method. It organizes feature vectors hierarchically, with fewer vectors at higher levels and more at lower ones. A probability function, $P(L_v) = F(L_v, l_m)$, determines the insertion level L_v based on a "level multiplier" l_m. After level generation, vectors are layered by their distances from the centroid, where closer vectors reside in layer 0 and farther ones in layer L. A bidirectional graph connects each vector to its M-Nearest Neighbor within the same layer and 1-Nearest Neighbor in subsequent layers. This results in layer 0 to L vectors having $M + L - 1$ to M neighboring nodes, impacting index size and recall. Typically, an optimal M ranges from 5 to 48, with larger values increasing index size and recall.

Algorithm 1: 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

$numLayer \leftarrow \lfloor \log_2 M \rfloor$
$cen \leftarrow$ mean of X
$dist \leftarrow$ distances from the centroid to all data vectors
$avg \leftarrow$ mean of all distances
$\sigma \leftarrow$ standard deviation of all distances
$u_b \leftarrow avg + f \times \sigma$
$l_b \leftarrow$ smallest of $dist$
$r \leftarrow \frac{u_b - l_b}{numLayer}$
$layeredElem \leftarrow \phi$
foreach (d, x) *of* $(dist, X)$ **do**
  $l \leftarrow \lfloor \frac{d}{r} \rfloor$
  add element x to layer l in $layeredElem$
end
return $layeredElem$

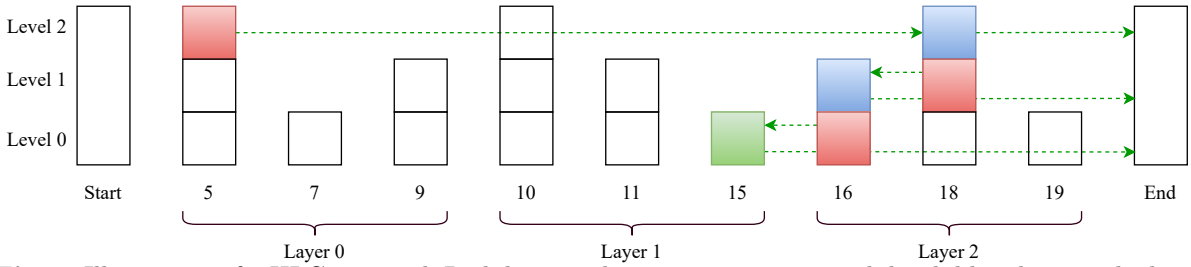

Fig. 3: Illustration of a HLG retrieval. Red denotes the starting point in each level, blue denotes the local optimum in each level, and green arrows show the direction of the greedy algorithm to the query(shown green).

Algorithm 2: 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 $graph \leftarrow \phi$
  foreach x *of* X **do**
2 | $graph \leftarrow$ ADD$(x, M, cand)$
3 **end**
4 $layeredElem \leftarrow$ LAYERING (X, M, f)
  $maxLayer \leftarrow size(layeredElem)$
  foreach $layer$ *of* $layeredElem$ **do**
5 | **if** $layer < maxLayer$ **then**
6 | | $clg \leftarrow$ get the graph for $layer$
  | | $nlg \leftarrow$ get the graph for $(layer+1)$
  | | **foreach** $elem$ *of* $layer$ **do**
7 | | | $n \leftarrow$ SEARCH$(nlg, elem, k = 1, cand = 1)$
  | | | update $graph$ inserting n to neighbor list of $elem$
8 | | **end**
9 | **end**
10 **end**

Indexing complexity The construction of the HLG index involves two main steps. Firstly, in the initial step (Algorithm 2, lines 2-4), each element undergoes iterative insertions to form a hierarchical proximity graph. The maximum number of levels in this graph is controlled by the parameter M. Utilizing an exponential decaying probability distribution, with m_l as $\frac{1}{log_2(M)}$, determines the maximum level for each element. Insertions start from the top level, with a greedy traversal of the graph to identify the closest neighbors of the inserted element x. This process repeats, using the closest neighbors from the previous level as starting points for the subsequent level search. This step's complexity is $O(N.log(N))$.

Next, we determine the layers of each element based on their distances from the centroid (Algorithm 1). An outlier filtering factor f is employed during this process to exclude elements that are f standard deviations from the mean distance. This ensures that outliers do not significantly affect the determination of layer boundaries. Subsequently, we extract the graphs for each layer from the constructed network.

In the second step (Algorithm 2, lines 9-12), for each element, we extract the closest neighbors from the upper layers and add them to the neighbor list in the previous graph, excluding elements in the last layer (Algorithm 2, line 8). This phase, being a series of greedy searches, also has a complexity of $O(N.log(N))$. Consequently, the overall complexity of HLG index building scales as $O(N.log(N))$. Neighboring nodes within the same layer aid HLG in improving recall, while those in upper layers expedite the search process. Hence, adjusting the connections between the inner and outer layers allows us to balance retrieval speed and recall.

HLG searching Figure 3 shows the k-NN retrieval process of the proposed Hierarchical Layered Graph (HLG) search method. Initially, the search begins at the upper level of the index, where edges are the longest, employing a greedy search to a local optimum. Subsequently, it progresses to lower levels, starting from the previous local optimum, until reaching the query and returning the top k-NN. Layering enables HLG to avoid exhaustive exploration within the same layer when

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
$L \leftarrow$ get highest level of g
for $l \in L, L-1, \ldots, 2$ *of* g **do**
2 $p \leftarrow$ extract nearest neighbor to q starting with ep
$ep \leftarrow p$
3 **end**
4 $C \leftarrow$ extract $cand$ neighbors to p at bottom level of g
$neighbors \leftarrow$ top k closest from C to q
return neighbors

the query resides in a different layer of the feature space. Additionally, it skips visiting nodes in distant layers from the current query node (Algorithm 3, Line $3-6$). The algorithm identifies the nearest neighbors for each inner layer using a greedy search, updating the network accordingly. Starting from the top level with the input network's entry point, it iterates to lower levels until reaching the second lowest level. Finally, at the network's bottom level (g), it retrieves the closest k neighbors to the query 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 of 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 $O\big(log\big(N_l\big)\big)$, 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 $O\big(log\big(N\big)\big)$.

Table 1: Experimental corpus for image deduplication.

Data	Train	Validation	Number of classes
cifar10 [27]	50,000	10,000	10
cifar100 [27]	50,000	10,000	100
Birds [28]	56,500	14,126	450
ImageNet [1]	309,032	77,259	1000

4 Experimental Setup

4.1 Dataset and Model Training

VIRD method is evaluated on three benchmark corpora: ImageNet [1], cifar10 and cifar100 [27], and one crowd-sensing iNaturalist-Birds corpus [28]. Table 1 shows the corpus characteristics, and each corpus comes with various classes showing the corpus's diversity, as shown in Figure 4. All the experiments are conducted on hardware configured with Ubuntu 20.04.3 server with 11th generation Intel® CoreTM i9-11900K @ 3.5GHzX16 CPU with 128GB RAM GPU NVIDIA Corporation GP102 [TITAN Xp] GPU Memory 12GB. We have chosen to use ResNet50[26] architecture for generating feature vectors for the image data. During each training epoch of the DNN model, a mini-batch consisting of 64 images is utilized for the Birds dataset. In comparison, a mini-batch comprising 128 images is employed for the remaining corpora. The learning rate is fixed at 0.0001, and each model is trained over 70 epochs. These configurations remain consistent across both the original and deduplicated datasets.

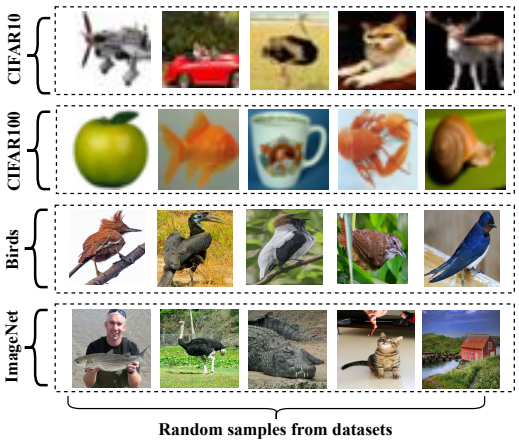

Fig. 4: Sample images from the experimental corpora showing the class diversity for each corpus.

4.2 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 measures the effectiveness of a deduplication method based on duplicate data elimination and the resulting mAP loss. Given the absence of ground truth for actual duplicates or near-duplicates in our experimental image corpora, to measure the effectiveness of comparing methods, we introduce a new metric termed *Deduplication efficiency* (DE):

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

The *Dedup* represents the percent of duplicate data elimination, where a higher value indicates a 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 corpus. It measures the total elapsed time from the start of the training process until the model has been fully trained.

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 corpus. It includes energy usage by computational hardware (such as CPUs or GPUs), memory storage devices, and associated cooling systems.

5 Experiments

Our work is closely aligned with CEDedup [29], which utilizes hash-based feature extraction techniques for efficient image deduplication. CEDedup inspires our work. 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 [30], PHash[31], DHash [32] and AHash [33] in terms of Deduplication efficiency.

The relationship between deduplication percentage and mAP based on the threshold T for the Cifar10 dataset is depicted in Figure 5 demonstrates that as T rises, the percentage of deduplication increases, leading to a decrease in the training data and consequently a gradual decline in mAP. When employing ResNet50 with the parameter configurations specified in section 4 and trained on the original Cifar10 dataset, we achieve an mAP of 95.40%. With T set to 2, VIRD removes 25.13% of redundant data, resulting in a comparable mAP of 95.03% for the Cifar10 corpus (Figure 5). However, determining the optimal

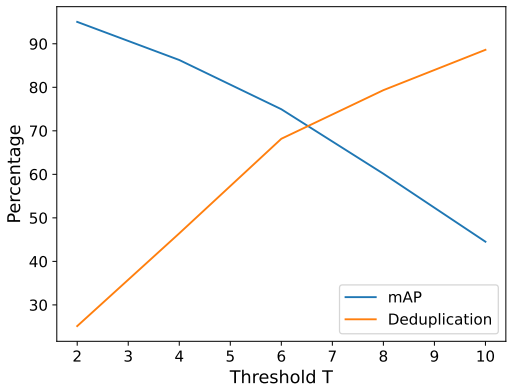

Fig. 5: mAP as a function of the threshold distance T for Cifar10 corpus.

Table 2: Training time and Energy consumption of Resnet50 model for original and deduplicated corpora

Dataset	Original		Deduplicated	
	Training time (minute)	Energy consumption (KWh)	Training time (minute)	Energy consumption (KWh)
Cifar10	153	0.92	111 ↓	0.67 ↓
Cifar100	188	1.14	144 ↓	0.87 ↓
Birds	1332	8.28	1106 ↓	6.88 ↓
ImageNet	326	1.99	278 ↓	1.62 ↓

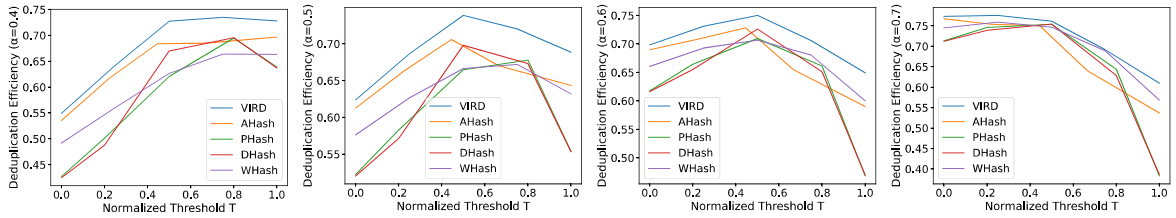

Fig. 6: mAP as a function of the weighting factor α for Cifar10 corpus.

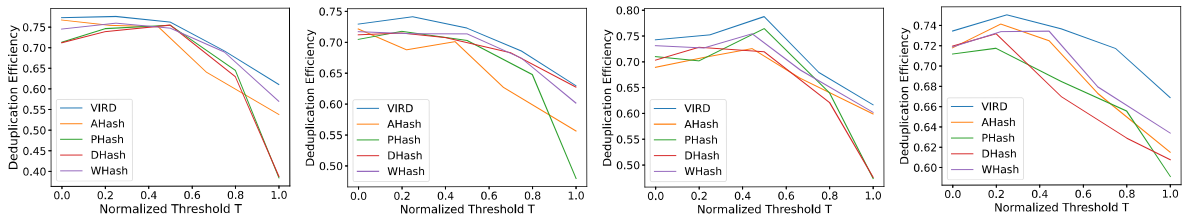

Fig. 7: Deduplication efficiency for comparing methods on Cifar10, Cifar100, Birds and ImageNet corpora at different threshold values.

threshold for DNN performance is challenging. Therefore, the Deduplication Efficiency (DE) metric aids in understanding the balance between data deduplication and mAP. The weighting factor α in DE signifies the relative importance of mAP loss to data deduplication, which can be manually set based on specific requirements. The α value greater than 0.5 indicates a higher emphasis on mAP. Figure 6 demonstrates the influence of α on deduplication efficiency across various values for the Cifar10 dataset. Given our objective of eliminating duplicates without compromising mAP, subsequent experiments are carried out with α set to 0.7, prioritizing the mAP. Figure 7 highlights VIRD's superior performance over comparing methods across all four experimental datasets in terms of DE. Performance is presented for normalized T values due to the variation in threshold distance across different methods.

The total training time and energy consumption of DNN on the original corpus and deduplicated corpus. We utilized a physical device, the P4400 p3 Kill A Watt meter (accurate within 0.2%), to measure the power consumption of each DNN model training. Using this device, we measured the power consumption of the CPU line. We conducted all the experiments for energy efficiency analysis with a similar setup specified in section 4.1. Table 2 shows the training time and total energy consumption during the Resnet50 training phase on the experimental corpora before and after the deduplication process. Here, we only present the results for Threshold distance T=2, where we observed the highest data deduplication with no more than 5% mAP loss. VIRD demonstrates reductions in total energy consumption by 27.17%, 23.68%, 18.59%, and 16.91%, respectively, for the Cifar10, Cifar100, Birds, and ImageNet datasets without any significant mAP loss. The findings support the effectiveness of VIRD in improving energy efficiency without compromising the overall performance of the DNN models. This green practice aligns with sustainability efforts by minimizing energy consumption in deep learning tasks, ultimately contributing to a more eco-friendly approach to data processing and model training.

6 Conclusion

Identifying and eliminating redundant images plays a crucial role in optimizing the Deep Neural Network (DNN) training process. The proposed VIRD framework presents a promising solution for optimizing the process of training DNNs by effectively identifying and removing redundant images from large-scale datasets. Through the implementation of graph-based approximate search and deduplication threshold adjustments, VIRD successfully balances the deduplication ratio and stability of mean Average Precision (mAP) while reducing the dataset size by 25.13%. Additionally, the streamlined training process facilitated

by VIRD leads to a significant reduction in energy consumption by 27.17%, making DNN training more practical and sustainable. These findings underscore the potential of VIRD in mitigating resource and computation consumption while ensuring the quality of data to enhance the accuracy of network training, making it a valuable framework for the field of visual processing and deep learning.

References

[1] 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, vol. 115, pp. 211–252 (2015). Springer

[2] Kaur, R., Bhattacharya, J., Chana, I.: Deep CNN-based online image deduplication technique for cloud storage system. Multimedia Tools and Applications, vol. 81, no. 28, pp. 40793–40826 (2022). Springer

[3] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y., et al.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning, vol. 2011, no. 5, pp. 7 (2011). Granada, Spain

[4] Nbt, Y., Ismail, A., Majid, N.A.A.: Deduplication image middleware detection comparison in standalone cloud database. Int. J. Adv. Comput. Sci. Technol. (IJACST), vol. 5, no. 3, pp. 12–18 (2016).

[5] Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green AI. Communications of the ACM, vol. 63, no. 12, pp. 54–63 (2020). ACM New York, NY, USA

[6] Xie, D., Zhang, L., Bai, L., et al.: Deep learning in visual computing and signal processing. Applied Computational Intelligence and Soft Computing, vol. 2017 (2017). Hindawi

[7] Ma, J., Jiang, X., Fan, A., Jiang, J., Yan, J.: Image matching from handcrafted to deep features: A survey. International Journal of Computer Vision, vol. 129, no. 1, pp. 23–79 (2021). Springer

[8] 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).

[9] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547 (2019). IEEE

[10] Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 824–836 (2018). IEEE

[11] Chen, Q., Zhao, B., Wang, H., Li, M., Liu, C., Li, Z., Yang, M., Wang, J.: Spann: Highly-efficient billion-scale approximate nearest neighborhood search. Advances in Neural Information Processing Systems, vol. 34, pp. 5199–5212 (2021).

[12] 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). IEEE

[13] Velmurugan, K., Baboo, L.D.S.S.: Content-based Image Retrieval using SURF and Color Moments. Global Journal of Computer Science and Technology, vol. 11, no. 10, pp. 1–4 (2011).

[14] 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), vol. 10, no. 4, pp. 1–15 (2014). ACM New York, NY, USA

[15] 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 (2009). IEEE

[16] Li, L., Zic, J.: Image Matching Algorithm based on Feature-point and DAISY Descriptor. Journal of Multimedia, vol. 9, no. 6, pp. 829–834 (2014). Citeseer

[17] 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 (2007). IEEE

[18] Chen, C.C., Hsieh, S.L.: Using binarization and hashing for efficient SIFT matching. Journal of Visual Communication and Image Representation, vol. 30, pp. 86–93 (2015). Elsevier

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

[20] 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 (2020). Springer

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

[22] Zhang, S.: Challenges in KNN classification. IEEE Transactions on Knowledge and Data Engineering, vol. 34, no. 10, pp. 4663–4675 (2021). IEEE

[23] Ukey, N., Yang, Z., Li, B., Zhang, G., Hu, Y., Zhang, W.: Survey on exact KNN queries over high-dimensional data space. Sensors, vol. 23, no. 2, pp. 629 (2023). MDPI

[24] 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). IEEE

[25] 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 (2022). IEEE

[26] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016).

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

[28] iNaturalist: iNaturalist is a joint initiative of the California Academy of Sciences and the National Geographic Society. (2022). https://www.inaturalist.org/observations

[29] 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 (2021). IEEE

[30] 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 (2017). IEEE

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

[32] Wang, J., Fu, X., Xiao, F., Tian, C.: DHash: Enabling Dynamic and Efficient Hash Tables. arXiv preprint arXiv:2006.00819 (2020).

[33] 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 (2018). Springer