EVIREC: Efficient Visual Indexing and Retrieval for Edge Crowd-sensing

***Abstract*—Due to the abundance of images in Visual crowd-sensing data, it is hard to identify and eliminate redundant data manually. In this paper, we propose Deep Indexing and Retrieval System for Efficient Use and Maintenance of Visual Crowdsensing on Edge (EVIREC) method that identifies and eliminates redundant data. EVIREC uses Deep Nural Network (DNN) to extract features from images and performs Approximate Nearest Neighbor (ANN) search to retrieve duplicate or near duplicate images in the database. Using a threshold distance in the feature space, EVIREC eliminates redundant images efficiently. The experimental results demonstrate that EVIREC surpasses both state-of-the-art hashing-based and CNN-based methods in terms of performance on the CIFAR10, CIFAR100, and Birds datasets. It successfully eliminates redundant images with an impressive rate of up to 99.99%, making it the most robust method for the image deduplication task in visual crowd-sensing image data.**

***Index Terms*—Data redundancy, crowd-sensing, Visual crowd-sensing, Big data, Mobile crowd-sensing**

I. Introduction

The rapid development of Internet of Things (IoT) devices has led to an explosion of data being generated from various sources. This data must be stored, processed, and analyzed to extract valuable insights and make informed decisions. One emerging paradigm addressing this challenge is Mobile crowd-sensing (MCS) [10], [25], which involves volunteers contributing data obtained by their sensor-enhanced mobile devices.

MCS leverages the capabilities of smartphones and other mobile devices to collect data from the environment. It can use different sensing modalities, including audio, pictures/videos, and numeric quantities such as GPS coordinates, temperature, and air quality. Among these modalities, visual crowd-sensing (VCS) has gained popularity. visual crowd-sensing utilizes the in-built cameras of intelligent devices, enabling users to contribute to various tasks by sending images or video information from their surroundings.

Compared to traditional approaches like mounting still cameras in specific locations, visual crowd-sensing provides more informative and diverse data. It allows for real-time data collection from different perspectives and locations, making it particularly useful for environmental monitoring, urban planning, and disaster management applications.

Several successful projects have been based on VCS, demonstrating its potential. Examples include CreekWatch [17], FlierMeet [8], PhotoCity [32], PhotoNet [33], WreckWatch [34], Mediascope [14]. These projects have shown that engaging a crowd of mobile device users makes it possible to gather a large volume of data from various sources, improving the coverage and accuracy of the collected information.

However, visual crowd-sensing also presents challenges, including the vast amount of data generated by multiple users contributing images and videos. Managing and processing this large-scale data becomes a significant task. Additionally, there may be data redundancy, where multiple users capture and contribute similar or identical information. Identifying and eliminating redundant data is crucial to reduce storage requirements, improve processing efficiency, and avoid duplication of efforts.

Addressing the issue of data redundancy in visual crowd-sensing requires efficient data management techniques. These techniques may involve data deduplication, where redundant data is identified and stored only once, or data aggregation methods to consolidate similar information from multiple sources. Advanced algorithms and machine learning techniques can efficiently analyze and process the data, identify patterns, extract relevant information, and summarize the crowd-sensing data.

The search for precise or nearly identical images has posed a significant challenge within a vast storage system. With the rise in VCS's popularity, users worldwide share many images. However, a considerable portion of these uploaded images is either slightly altered or exact replicas of an original image [22]. Consequently, a massive database of duplicate or near-duplicate images rises within the storage system [26]. This abundance of duplicate images negatively impacts the performance of the image storage system and drives up its costs. Furthermore, efficiently indexing or retrieving images from an extensive image cloud storage system proves highly challenging [16]. Therefore, there is a need for an efficient real-time or online technique that can filter out duplicates of an original image before sending it to the storage, thus enhancing overall system storage efficiency.

In this paper, we propose a novel approach called **D**eep **I**ndexing and **R**etrieval **S**ystem for Efficient Use and Maintenance of **V**isual **C**rowdsensing on **E**dge (EVIREC) to address the issue of redundant data in Visual crowd-sensing (VCS) image data. EVIREC aims to efficiently identify and eliminate duplicate or near duplicate images to the previously stored image data from the visual crowd-sensing data.

The suggested approach entails extracting and storing features from newly arrived images incrementally. These features serve as descriptive representations of the images, capturing crucial visual attributes. By incrementally extracting and preserving these features, we establish a database of pre-existing

features and index them in a way that allows for efficient searching. This enables us to determine the similarity between an incoming image and the previously stored images.

To determine the similarity between an incoming image and the existing features in the database, we propose Hierarchical Layered Graph (HLG), an Approximate Nearest Neighbor (ANN) search to retrieve the most similar item from the feature database. ANN search is chosen over the traditional k-Nearest Neighbor (k-NN) approach because it can handle large-scale databases. ANN search is more suitable for visual crowd-sensing large databases consisting of many images, as it can handle searching in a million-scale feature database. In contrast, k-NN can become very slow when dealing with high-dimensional features and large databases [18].

After retrieving the most similar feature using the HLG search, EVIREC employs either the Euclidean distance in the feature space or the log similarity score to determine the similarity of an incoming image. If the incoming image is deemed sufficiently similar to the retrieved feature, it is classified as redundant and can be discarded. Conversely, if the image is distinctive enough, it is considered unique. The feature is included in the existing feature database, and the incoming image is stored in the image database. EVIREC reduces the storage requirements and processing overhead associated with redundant data in visual crowd-sensing big data. By eliminating duplicate or near duplicate images, the dataset becomes more compact and efficient, improving the overall performance of visual crowd-sensing applications. In summary, we make the following contributions-

1) We propose the EVIREC framework, which, to our knowledge, is the first method that utilizes graph-based approximate nearest neighbor search in deep features to accomplish the image de-duplication task.
2) We propose a novel graph-based approximate nearest neighbor indexing and search method, Hierarchical Layered Graph (HLG), in an extensive deep descriptor database to efficiently retrieve the most similar image descriptor.
3) We employ log similarity to determine whether a query image closely resembles any image in the database.

In section III, we discuss our proposed EVIREC method in detail, and in section V, we show our experimental results.

II. Related Work

Data selection and redundancy elimination are the critical challenges in visual crowd-sensing data [9]. Several image de-duplication methods have been proposed over the past few years, which can be categorized mainly into Hashing-based and Convolutional Neural Network (CNN) based methods.

This idea is easy and uncomplicated: If more people report an observation, it is more likely to be significant, whereas things with few observers can be viewed as outliers. Data utility or usefulness in TaskMe [7] is measured by predefined task restrictions rather than clustering findings. A photo is considered an outlier in PhotoNet [33] if it is geographically close to a famous picture cluster yet visually distinct from the group. However, lone images are sometimes relevant, and the pertinent sensing targets might be found in areas with few observers. EVIREC handles this problem by keeping the lone images if they differ in features from other images in the database. The creation of data selection strategies for visual crowd-sensing has been extensively studied. For instance, CrowdPic [6] presents a general picture collection framework that facilitates effective picture grouping and redundancy removal based on multi-dimensional job constraints. To support online crowdsourcing photo grouping and represent the task requirements, the pyramid-tree (PTree) method is presented. Some visual crowd-sensing programs try to acquire data selection techniques from human experience or expert knowledge. The computational domain-specific filming rules in MoVieUp [35] include the 30-degree rule and the less shot-switching concept (there should be at least 30 degrees' variation between shooting angles to prevent jump cuts in camera selection). A framework called MoViMash [28] compiles video clips of an event shot from various distances and viewpoints. They used a hidden Markov model to learn from the experiences of expert editors, such as choosing the shooting angle and distance, shot length, and transitions.

A. Hashing-based

Image hashing involves examining the content of an image and generating a distinct identifier based on its specific characteristics. The image is processed through a hash function, resulting in a value representing its visual properties. Similar images should produce similar hash values, and by comparing the variances in hashes, visually similar images are identified.

QHash [24] was explicitly designed for low-variance image de-duplication which aims to minimize the occurrence of false positives and false negatives during the de-duplication process, thereby improving the overall accuracy and efficiency of image de-duplication systems. The algorithm consists of three main steps: preprocessing, feature extraction, and hash code generation. In the preprocessing step, the input image is resized and converted to grayscale for efficiency. Feature extraction involves applying quantization and salient point detection algorithms to capture global and local features. The hash code generation combines the extracted features with a hash function to produce the final hash code.

The AHash algorithm [4] downsamples an image to a size of N × N pixels (8 × 8 in this paper) to remove high frequencies and details. After that, the RGB image is converted to grayscale, and the average pixel value Pmean is determined. Each pixel Pval is then compared to Pmean, and a bit value Pbit of 1 is assigned if Pval is greater than Pmean; otherwise, 0. The resulting 8 × 8 binary matrix is flattened into a 64-bit integer to produce the final hash. AHash is a fast and straightforward hashing algorithm. However, directly comparing with the mean might not provide consistent outcomes.

The initial step of the Difference Hashing (DHash) algorithm [] involves resizing the image to a resolution of N × (N + 1) (8 × 9 in this particular paper). After that, the RGB image is

converted to grayscale. To convert the pixels into binary form, DHash examines the neighboring pixels. In detail, each pixel is assigned a value of 1 if its pixel value is greater than the pixel to its right; otherwise, it is assigned a value of 0. As a result of this process, an 8×8 binary matrix is created. The DHash algorithm improves upon AHash by comparing adjacent pairs of pixels, thereby preserving more local patterns.

The Perceptual Hashing (PHash) algorithm [38] involves resizing the image to a resolution of 32×32 pixels. Then, the algorithm utilizes the Discrete Cosine Transform (DCT) [1] to convert the spatial RGB values into a set of frequencies and magnitudes. Only the top-left 2-dimensional matrix measuring 8×8 is retained to eliminate the high-frequency elements. The average value of this matrix is calculated to create a binary matrix, which is subsequently flattened into a 64-bit integer serving as the hash representation.

The Wavelet Hashing (WHash) algorithm [30] shares similarities with PHash, with the key difference being its utilization of the Discrete Wavelet Transform (DWT) [29] for the conversion of spatial RGB values into frequencies and magnitudes.

B. Deep Neural Network Modeling

A deep CNN model is trained using a large dataset of diverse images to learn and extract discriminative features [16]. The model is trained to map similar images to close feature representations while maintaining a significant distance for dissimilar images. During the online de-duplication phase, when a new image is uploaded to the cloud system, it undergoes a series of pre-processing steps. These steps involve resizing, normalization, and feature extraction using the pre-trained deep CNN model. The extracted features are then compared with the features of existing images in the cloud storage. To efficiently search for duplicate images, an indexing mechanism based on locality-sensitive hashing (LSH) is employed. LSH enables fast approximate nearest neighbor search, quickly identifying potential duplicates.

Cost-effective convolutional neural nets training based on image deduplication (CE-Dedup) [23] focuses on assessing the impact of near-duplicate images on CNN training performance. CE-Dedup combines a hashing-based image de-duplication approach with downstream CNN-based image classification tasks. The framework aims to balance a high de-duplication ratio with maintaining stable accuracy. This is achieved by heuristically adjusting the de-duplication threshold. The goal of CE-Dedup is to make CNN training more cost-effective by reducing redundancy in the training data without sacrificing performance.

A technique was proposed by S. Thaiyalnayaki et al. [31] to identify near-duplicate images by utilizing the Speeded-Up Robust Feature (SURF) algorithm and the segmented minhash algorithm. The SURF algorithm is employed to extract image features, while the segmented minhash algorithm is used to index the similarity of the extracted images. Locality Sensitive Hashing is applied to index the near-duplicate images. The paper [5] introduced a rapid image retrieval method that involves converting 128-DSIFT features into 128-bit binary representations. Hash values are computed for each feature, and the hamming distance is utilized to identify images with similarities. This technique significantly decreases the search retrieval time. The framework [37] consists of three main components: feature extraction, similarity measurement, and duplicate image detection. In the feature extraction stage, various techniques such as scale-invariant feature transform (SIFT), SURF, and deep learning-based methods, are employed to extract distinctive features from images. *TODO: what is so special about it? – Toufik*

EVIREC handles the camera angle and orientation problem by training a DNN model with varying images in different orientations, lighting, and angles. The DNN model can capture these varying conditions and extract features from an image database.

III. Methodology

EVIREC works both for detecting and eliminating redundant images in an existing database and building a database without adding redundant images. Fig.1 shows the proposed pipeline of our EVIREC method. There are two main phases, Feature extraction using DNN and ANN search, which we discuss later in detail. The whole process is done incrementally by scanning one image at a time. Each image is passed to the DNN for the feature extraction and the feature is saved to the feature database. Then the features are indexed using the Hierarchical Layered Graph (HLG), an ANN searching method for searching for the most similar image feature. During the ANN Search using HLG, we retrieve the most similar image feature from the feature database to an incoming image feature. Based on the distance from the incoming image feature to the retrieved image feature, we decide whether an incoming image is redundant. We have used Euclidean distance as the similarity metric in our proposed method. The distance threshold can be varied to decide the redundancy of an image. For example, if the threshold value is set to a lower distance, then fewer images will be eliminated from the database. Setting the threshold to a higher distance will eliminate more images from the database.

A. Feature extraction with Deep Neural Network (DNN)

The first step toward eliminating data redundancy in the crowd-sensing setting is to represent the data efficiently. From the above dataset section as we can see that the input data in our pipeline is images. A very common method of representing image data is in vector format. Many methods are devised to represent the image data in vector format; among them, using DNN to extract feature vectors is the most popular. 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 [2] for object detection in images, we have chosen to use ResNet50 architecture for generating feature vectors for the image data. From Figure 1, it can be seen that the ResNet50 model is built upon many Convolution(Conv) blocks stacked one after one. The first seven blocks in the network are ConV blocks with 64 channel

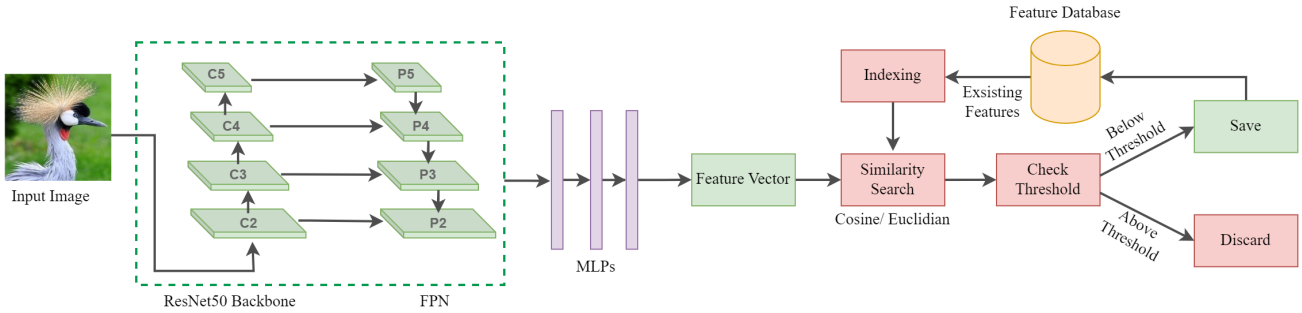

Fig. 1. Proposed EVIREC pipeline.

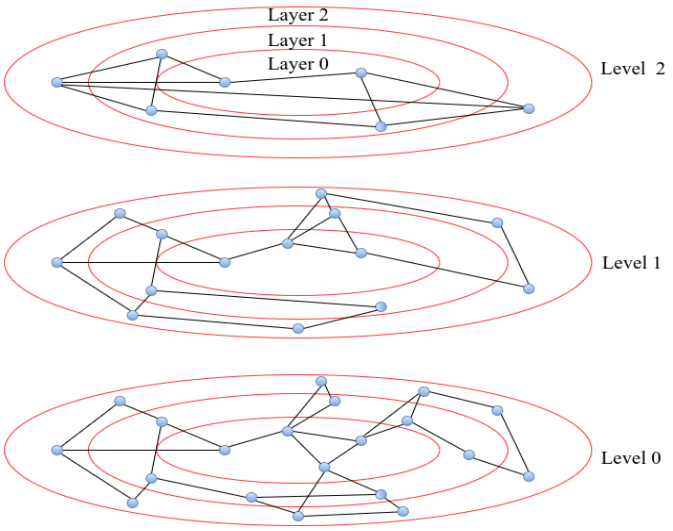

Fig. 2. Illustration of a Hierarchical Layered Graph (HLG) indexing. Each feature vector connects to its M-Nearest Neighbor within the same layer and 1-Nearest Neighbor in the next layers.

outputs and only one stride at the beginning. Then Next block starts with a Conv block with a stride of 2 and an output channel of 128. This CNN fashion follows onward with 256 and 512 output channels. Next, we perform average pooling on the last Conv layer output. Finally, we feed the output from the average pool into a linear layer and save the output from this layer as a feature in our database in 512 lengths of a vector.

B. Approximate Nearest Neighbor (ANN) search using Hierarchical Layered Graph (HLG)

a) ***Index building****:* Fig. 2 illustrates the index structure of our proposed Hierarchical Layered Graph (HLG) approach. Hierarchical Layered Graph (HLG) first arranges all feature vectors in 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, as more vectors are moved to the level 0, increases the average number of search traversals. After generating the levels, the Hierarchical Layered Graph (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 where a larger value of M leads to a larger index size and higher recall.

b) ***k-Nearest Neighbor (k-NN) retrieval****:* Fig. 3 shows the k-NN retrival 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 (Fig. 3). The search then switches to the lower level, where the edges are shorter. 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 Hierarchical Layered Graph (HLG) avoid visiting all neighboring nodes within the same layer if the query is in a different layer in the feature space. Moreover, the Hierarchical Layered Graph (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).

The index construction outlined in the Algorithm 1 has two phases: (1) building a hierarchical level of proximity graphs within the same layer and (2) adding next-layer connections. The exponential decaying probability distribution $(\lfloor -log_2(unif(0, ..., 1) \times m_l) \rfloor)$ determines the maximum level for each element, where m_l is $\frac{1}{log_2(M)}$. Therefore, the maximum number of levels in the hierarchical graph can be controlled by the maximum established connection parameter

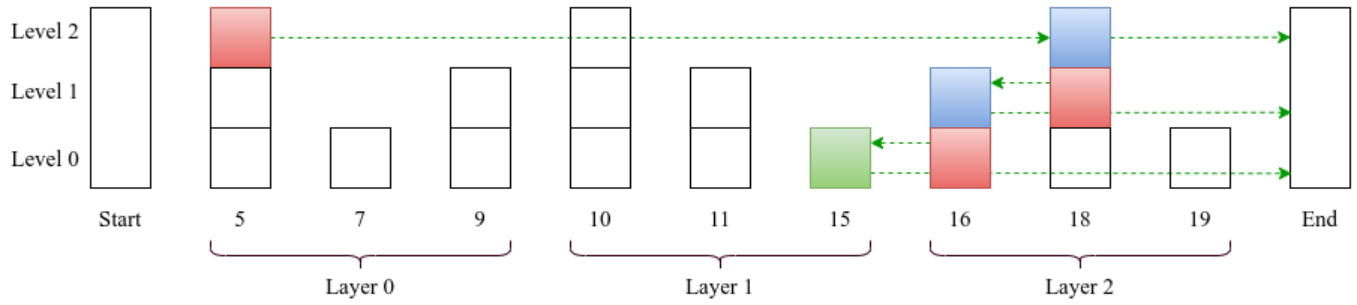

Fig. 3. Illustration of a Hierarchical Layered Graph (HLG) retrieval. Red denotes the starting point in each level, blue denotes the local optimum in each level, and green arrows show the direction of the greedy algorithm to the query(shown green).

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 in the previous level as entry points for the algorithm to carry on with the search from the subsequent level. The greedy search algorithm in Algorithm 3 is used to find the closest neighbors, which we discuss later in this section.

Algorithm 1: BUILD$(HLG, X, M, cand, f)$

Input: hierarchical layered graph *HLG*, data vector *X*, number of established connections *M*, size of dynamic candidate list *cand*, outlier filtering factor *f*

Output: UpdateHierarchical Layered Graph (HLG) inserting all elements

$graph \leftarrow \phi$
foreach x *of* X **do**
  $graph \leftarrow$ ADD$(x, M, cand)$
end
$layeredElem \leftarrow$ LAYERING (X, M, f) //Algorithm 2
foreach $layer$ *of* $layeredElem$ **do**
  $clg \leftarrow$ get the graph for $layer$
  $nlg \leftarrow$ get the graph for $(layer + 1)$
  foreach $elem$ *of* $layer$ **do**
    $n \leftarrow$ SEARCH$(nlg, elem, k = 1, cand = 1)$
    update $graph$ inserting n to neighbor list of $elem$
  end
end

In the next phase of index construction, we determine the layers of each element based on their distances from the centroid (Algorithm 2). 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. For each inner layer, we identify the closest nearest neighbor in the subsequent layers using the greedy search algorithm (Algorithm 3). Finally, we update the previous network by adding the closest neighbor found in the subsequent layers.

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

$numLayer \leftarrow \log_2 M$
$cen \leftarrow$ mean of X
$dist \leftarrow$ distances from 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**
  $l \leftarrow \frac{d}{r}$
  add element x to layer l in $layeredElem$
end
return $layeredElem$

The greedy search process (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 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 Hierarchical Layered Graph (HLG) index level is built as a navigable small-world graph, allowing the greedy search path's hop count to scale logarithmically. Hierarchical Layered Graph (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

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

```
ep ← get entry point of g
L ← get highest level of g
for l ∈ L, L − 1, . . . , 2 of g do
    p ← extract nearest neighbor to q starting with ep
    ep ← p
end
C ← extract cand neighbors to p at bottom level of g
neighbors ← top k closest from C to q
return neighbors
```

of the items on the greedy path is multiplied to get the overall amount of distance calculations. As a result, each level of the Hierarchical Layered Graph (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 Hierarchical Layered Graph (HLG) is determined by $O\big(log\big(N\big)\big)$.

Index building complexity The Hierarchical Layered Graph (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 ANN searches at different levels. Thus, the first phase has a complexity of $O\big(N.log\big(N\big)\big)$. The second phase of Hierarchical Layered Graph (HLG) index building is also a series of ANN searches at different layers. Thus, similar to the first phase, the second phase has a complexity of $O\big(N.log\big(N\big)\big)$. Therefore, the overall complexity of the index building of the Hierarchical Layered Graph (HLG) scales as $O\big(N.log\big(N\big)\big)$.

C. Similarity Metrics

We have used two different measurement techniques to evaluate our proposed de-duplication method. As with other previous works, we use traditional Euclidian distance-based similarity search. Moreover, we use cosine similarities as a new way of similarity finding among feature vectors leveraging the idea from contrastive learning [11].

Normalized cosine similarity: is a simple process of measuring pair-to-pair relationships based on the similarities between different pairs, such as query-negative samples. We leverage the idea of feature discrimination calculation using cosine similarities from contrastive learning and integrate it into our pipeline for similarity calculation. Many versions of contrastive learning are available for feature representation learning [12], [27], [36]. Previous works [3], [15] used cosine similarity with Informative Noise Contrastive Estimation (InfoNCE) [27] and successfully discriminated vectors in feature space. Motivated by this, we also use the normalized cosine similarity because of its simplicity and faster finding of dissimilarities between features.

$$sim(u, v) = u^T/(||u|| * ||v||) \tag{1}$$

$$x = x/max(||x||_p, \epsilon) \tag{2}$$

The formula for calculating the normalized cosine similarity score is presented in Eq. 1. Here, *Query* is the feature vector from the query image. On the other hand, *Negative* features are all the other feature vectors saved in the server database. Before performing the cosine similarity calculation, we normalize any vector x using Eq. 2. The Eq. 1 captures the similarity of two features u and v. The output ranges from 0 to 1, where 0 denotes no similarity and 1 denotes very high similarity.

IV. Dataset

To evaluate the performance of our proposed EVIREC method, we have used two publicly available data sets, CIFAR10 and CIFAR100 [19], and one crowd-sensing data set collected from *iNaturalist* [13] online database. Each dataset comes with various classes showing the dataset's diversity.

CIFAR10: Our first experimental dataset is a very well-known benchmark dataset *CIFAR10*, for computer vision task. There are a total of 60,000 color images in the dataset, each of which is a 32×32 pixel resolution. The dataset is divided into ten classes, each containing 6,000 images. The classes include common objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. We split the dataset into train and test sets. The train set contains 40,000 images, and the test set contains 20,000 images, where all classes are equally distributed.

CIFAR100: The CIFAR-100 dataset is a widely used benchmark dataset in data deduplication. CIFAR-100 consists of 100 classes, with each class containing 600 images. Moreover, these categories are divided into 20 superclasses, each containing five classes. The super classes capture higher-level semantic information, while the individual classes represent specific objects or characteristics. The dataset contains 50,000 training and 10,000 test images, equally distributed across different classes. Each color image in CIFAR-100 has a 32×32 pixels resolution. We choose this dataset due to its challenges due to its relatively low-resolution images and the presence of fine-grained classes. de-duplication becomes harder as many classes share visual similarities, requiring models to learn subtle discriminative features.

iNaturalist-Birds: Our last experimental dataset is from the iNaturalist. The iNaturalist Birds dataset is a comprehensive collection of bird species images prepared for bird classification and identification. It is derived from the iNaturalist platform, a popular online community for nature enthusiasts to share observations and photographs of various species. The iNaturalist Birds dataset contains many bird images covering many species worldwide. We have chosen 450 different bird classes with equal distribution for the de-duplication task.

The inter-class similarities and intra-class dissimilarities make the dataset perfect for exploring de-duplication tasks. We have 73,776 color images in the dataset, with dimensions of 224×224 pixels. The training set contains 70626 images and we perform the deduplication task on the training set for each method.

TABLE I
EXPERIMENTAL DATA SET FOR IMAGE DEDUPLICATION.

Data	Number of images	Size (MB)	Number of classes
CIFAR10 [20]	5000	47.4	10
CIFAR100 [21]	5000	114.2	100
Birds	70626	1600	450

V. EXPERIMENT

We compare the performance of our EVIREC methods with hashing-based methods and CNN-based methods. The evaluation is conducted using three metrics:

Number of duplicates found (Dup): This metric measures the ability of each method to identify and detect duplicate images within a dataset. Duplicate images refer to multiple copies or instances of the same image. The higher the number of duplicates found, the better the method's performance in identifying and flagging redundant images.

Database size reduction in megabytes (δ_s): This metric quantifies the reduction in storage space achieved by applying each method to the dataset. It measures the difference in the size of the original dataset and the size of the dataset after applying the method. A larger reduction in database size indicates better efficiency in terms of storage requirements.

Percentage of redundancy elimination (ξ): This metric assesses the extent to which each method eliminates redundancy within the dataset. Redundant images are those that provide no additional information compared to other images in the dataset. The percentage of redundancy elimination is calculated using the equation referenced as Eq. 3, where R_t represents the total number of redundant images in the dataset.

To calculate the percentage of redundancy elimination, the total number of images in the dataset is subtracted from the number of classes in that dataset. This subtraction yields the number of redundant images, as classes represent unique categories, and any additional instances of images within the same class are considered redundant. Consequently, the optimal number of unique images in any dataset equals the number of classes present in that dataset.

Overall, the comparisons aim to evaluate the performance of EVIREC method in terms of its ability to identify duplicates, reduce database size, and eliminate redundancy in comparison to other hashing-based and CNN-based methods. The metrics provide quantitative measures to assess the effectiveness of each method, and the percentage of redundancy elimination specifically highlights the value of unique images within a dataset based on the number of classes. All the useful notation used in this section are enlisted in Table II.

$$\xi = \frac{Dup}{R_t} \times 100\% \tag{3}$$

TABLE II
NOTATION TABLE.

Symbol	Description
Th_d	Euclidean distance threshold
Th_s	Cosine similarity threshold
Dup	Number of duplicate or near duplicate images
δ_s	Database size reduction in megabytes
ξ	Percentage of redundancy elimination
R_t	Number of redundant images in the data set

A. Comparison with hashing-based methods

TABLE III
DE-DUPLICATION RESULTS ON THE CIFAR10 DATASET WITH 50000 IMAGES AND TOTAL SIZE OF 47.4 MB.

Method	Th_d	Dup	δ_s	ξ
PHash	10	2292	1.97	4.58%
PHash	15	22128	19.26	44.26%
PHash	20	48985	43.00	97.99%
DHash	10	2088	1.81	4.17%
DHash	15	27874	24.37	55.75%
DHash	20	48541	42.61	97.10%
WHash	10	31260	27.18	62.53%
WHash	15	44297	38.78	88.61%
WHash	20	49617	43.56	99.25%
AHash	10	29722	25.81	59.45%
AHash	15	45458	39.82	90.93%
AHash	20	49509	43.46	99.04%
EVIREC	10	44297	38.78	88.61%
EVIREC	15	49235	46.71	98.49%
EVIREC	**20**	**49857**	**47.27**	**99.73%**

TABLE IV
DE-DUPLICATION RESULTS ON THE CIFAR100 DATASET WITH 50000 IMAGES AND TOTAL SIZE OF 114.2 MB.

Method	Th_d	Dup	δ_s	ξ
PHash	10	1817	3.63	3.64%
PHash	15	19811	41.36	39.70%
PHash	20	48933	104.11	98.06%
DHash	10	4313	8.46	8.64%
DHash	15	30517	63.96	61.15%
DHash	20	48650	103.42	97.49%
WHash	10	31494	64.98	63.11%
WHash	15	43975	92.64	88.12%
WHash	20	49547	105.33	99.29%
AHash	10	29818	61.61	59.75%
AHash	15	45063	95.12	90.30%
AHash	20	49458	105.13	99.11%
EVIREC	10	43993	92.65	88.13%
EVIREC	15	49551	105.34	99.30%
EVIREC	**20**	**49744**	**114.07**	**99.69%**

We conduct a comparative analysis of our EVIREC method against four state-of-the-art hashing-based methods, namely WHash [30], PHash [38], DHash [] and AHash [4] for the image deduplication task. To evaluate each method, we utilize

TABLE V
DE-DUPLICATION RESULTS ON THE BIRDS DATASET WITH 70626 IMAGES AND TOTAL SIZE OF 1600 MB.

Method	Th_d	Dup	δ_s	ξ
PHash	10	5826	120.38	8.30%
PHash	15	37365	787.56	53.24%
PHash	20	69558	1490.6	99.11%
DHash	10	3683	77.26	5.24%
DHash	15	44437	945.54	63.32%
DHash	20	69123	1481.51	98.50%
WHash	10	34716	727.45	49.47%
WHash	15	59241	1261.10	84.42%
WHash	20	70002	1500.12	99.75%
AHash	10	33244	692.34	47.37%
AHash	15	61781	1317.24	88.03%
AHash	20	69839	1496.70	99.52%
EVIREC	10	59241	1261.10	84.42%
EVIREC	15	68495	1550.3	97.60%
EVIREC	**20**	**69982**	**1554.7**	**99.81%**

three different distance thresholds ($Th_d = [10, 15, 20]$) and apply them to experimental datasets. The performance of the comparing methods on CIFAR10, CIFAR100, and Birds datasets is presented in Table III, Table IV, and Table V, respectively.

The results from the experiment conducted on CIFAR10 (see Table III) demonstrate that EVIREC successfully identifies 49,857 duplicate images out of a total of 49,990 images, resulting in a redundancy elimination rate of 99.73% when using a distance threshold of $Th_d = 20$. Moreover, EVIREC effectively reduces the size of the database by 47.27 megabytes, compared to the original size of 47.4 megabytes. WHash, a competing method, achieves similar results to EVIREC by detecting 49,857 duplicate images, leading to a redundancy elimination rate of 99.25% and reducing the database size by 43.56 megabytes. On the other hand, PHash, DHash, and AHash demonstrate redundancy elimination rates of 97.99%, 97.10%, and 99.04%, respectively.

When considering other distance thresholds, EVIREC outperforms the hashing-based methods listed in Table III. The same trend is observed for the CIFAR100 dataset, as shown in Table IV, where EVIREC successfully detects 49,744 duplicate images, resulting in a redundancy elimination rate of 99.69%. This leads to a reduction in the database size by 114.07 megabytes out of the original 114.2 megabytes. Additionally, Table V illustrates the effectiveness of EVIREC on the Birds dataset, detecting 69,982 duplicate images and achieving a redundancy elimination rate of 99.81%. This results in a database size reduction of 1554.7 megabytes out of the original 1600 megabytes.

B. Comparison with CNN-based methods

We have conducted a comparative analysis between our EVIREC method and the state-of-the-art CNN-based method CE-Dedup [23]. The evaluation of our method's performance was based on the cosine similarity threshold. The experimental results for CIFAR10, CIFAR100, and the Birds dataset are presented in Table VI, Table VII, and Table VIII respectively, considering three different similarity thresholds ($Th_s = [0.85, 0.90, 0.95]$).

Across all three datasets and corresponding similarity thresholds, EVIREC consistently outperforms CE-Dedup. For the CIFAR10 dataset, EVIREC successfully detects 49,988 duplicate images out of a total of 49,990 images, resulting in a redundancy elimination rate of 99.99% and reducing the database size by 47.39 megabytes out of the original 47.4 megabytes for $Th_s = 0.85$ (Table VI). In contrast, CE-Dedup only identifies 8,638 duplicate images out of 49,990, achieving a redundancy elimination rate of 17.27% and reducing the database size by 7.59 megabytes for the same similarity threshold (Table VI). Similar trends can be observed in Table VII for the CIFAR100 dataset and Table VIII for the Birds dataset. CE-Dedup eliminates 65.96% of redundant images in the Birds dataset, while EVIREC achieves a significantly higher redundancy elimination rate of 97.73% (Table VIII). Based on the experimental results, it is evident that EVIREC consistently outperforms CE-Dedup across all three datasets (Table VI, VII, VIII).

Therefore, EVIREC has proven to be the most robust method for the image deduplication task when compared to state-of-the-art hashing-based and CNN-based methods.

TABLE VI
DE-DUPLICATION RESULTS ON THE CIFAR10 DATASET WITH 50000 IMAGES AND TOTAL SIZE OF 47.4 MB.

Method	Th_s	Dup	δ_s	ξ
CEDedup	0.85	8638	7.59	17.27%
CEDedup	0.90	1864	1.64	3.73%
CEDedup	0.95	297	0.26	0.59%
EVIREC	**0.85**	**49988**	**47.39**	**99.99%**
EVIREC	0.90	49944	46.35	99.90%
EVIREC	0.95	47957	42.35	96.11%

TABLE VII
DE-DUPLICATION RESULTS ON THE CIFAR100 DATASET WITH 50000 IMAGES AND TOTAL SIZE OF 114.2 MB.

Method	Th_s	Dup	δ_s	ξ
CEDedup	0.85	9389	19.39	18.81%
CEDedup	0.90	3224	6.56	6.46%
CEDedup	0.95	941	1.97	1.88%
EVIREC	**0.85**	**49887**	**114.02**	**99.97%**
EVIREC	0.90	49835	113.85	99.87%
EVIREC	0.95	47519	105.45	95.23%

VI. CONCLUSION

Visual crowd-sensing (VCS) asks users to contribute to different tasks by sending images or video information from their surroundings. Due to the abundance of participant data, it is hard to identify and eliminate redundant data from the image database. Our proposed Data Redundancy Identification and Elimination (EVIREC) method extracts features from an image database using Deep Neural Network (DNN). It performs Approximate Nearest Neighbor (ANN) search to

TABLE VIII
DE-DUPLICATION RESULTS ON THE BIRDS DATASET WITH 70626 IMAGES AND TOTAL SIZE OF 1600 MB.

Method	Th_s	Dup	δ_s	ξ
CEDedup	0.85	46291	984.00	65.96%
CEDedup	0.90	18447	389.72	26.28%
CEDedup	0.95	4769	103.75	6.79%
EVIREC	**0.85**	**68587**	**1554.3**	**97.73%**
EVIREC	0.90	67968	1539.9	96.85%
EVIREC	0.95	67849	1537.5	96.68%

retrieve similar images. EVIREC then efficiently eliminates the redundant data based on the distant threshold in the feature space. The experimental result shows that EVIREC outperforms state-of-the-art hashing-based and CNN-based methods for the CIFAR10,CIFAR100 and Birds dataset by eliminating redundant images upto 99.99%.

REFERENCES

[1] Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE transactions on Computers **100**(1), 90–93 (1974)

[2] Biswas, D., Rahman, M.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

[3] Biswas, D., Tešić, J.: Progressive domain adaptation with contrastive learning for object detection in the satellite imagery (2023)

[4] 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. The PAAMS Collection-15th International Conference, PAAMS 2017 15. pp. 183–190. Springer (2018)

[5] 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)

[6] Chen, H., Guo, B., Yu, Z., Chen, L., Ma, X.: A generic framework for constraint-driven data selection in mobile crowd photographing. IEEE Internet of Things Journal **4**(1), 284–296 (2017)

[7] Guo, B., Chen, H., Yu, Z., Nan, W., Xie, X., Zhang, D., Zhou, X.: Taskme: Toward a dynamic and quality-enhanced incentive mechanism for mobile crowd sensing. International Journal of Human-Computer Studies **102**, 14–26 (2017)

[8] Guo, B., Chen, H., Yu, Z., Xie, X., Huangfu, S., Zhang, D.: Fliermeet: a mobile crowdsensing system for cross-space public information reposting, tagging, and sharing. IEEE Transactions on Mobile Computing **14**(10), 2020–2033 (2014)

[9] Guo, B., Han, Q., Chen, H., Shangguan, L., Zhou, Z., Yu, Z.: The emergence of visual crowdsensing: Challenges and opportunities. IEEE Communications Surveys & Tutorials **19**(4), 2526–2543 (2017)

[10] Guo, B., Wang, Z., Yu, Z., Wang, Y., Yen, N.Y., Huang, R., Zhou, X.: Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm. ACM computing surveys (CSUR) **48**(1), 1–31 (2015)

[11] Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 1735–1742. IEEE (2006)

[12] Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. arXiv preprint arXiv:1808.06670 (2018)

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

[14] Jiang, Y., Xu, X., Terlecky, P., Abdelzaher, T., Bar-Noy, A., Govindan, R.: Mediascope: selective on-demand media retrieval from mobile devices. In: Proceedings of the 12th international conference on Information processing in sensor networks. pp. 289–300 (2013)

[15] Kang, G., Jiang, L., Yang, Y., Hauptmann, A.G.: Contrastive adaptation network for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4893–4902 (2019)

[16] 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)

[17] Kim, S., Robson, C., Zimmerman, T., Pierce, J., Haber, E.M.: Creek watch: pairing usefulness and usability for successful citizen science. In: Proceedings of the SIGCHI conference on human factors in computing systems. pp. 2125–2134 (2011)

[18] 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)

[19] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[20] Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 (canadian institute for advanced research) http://www.cs.toronto.edu/ kriz/cifar.html

[21] Krizhevsky, A., Nair, V., Hinton, G.: Cifar-100 (canadian institute for advanced research) http://www.cs.toronto.edu/ kriz/cifar.html

[22] Li, J., Qian, X., Li, Q., Zhao, Y., Wang, L., Tang, Y.Y.: Mining near duplicate image groups. Multimedia Tools and Applications **74**(2), 655–669 (2015)

[23] Li, X., Chang, L., Liu, X.: Ce-dedup: Cost-effective convolutional neural nets training based on image deduplication. In: 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). pp. 11–18. IEEE (2021)

[24] Li, X., Chang, L., Liu, X.: Qhash: An efficient hashing algorithm for low-variance image deduplication. In: 2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys). pp. 9–15. IEEE (2021)

[25] Ma, H., Zhao, D., Yuan, P.: Opportunities in mobile crowd sensing. IEEE Communications Magazine **52**(8), 29–35 (2014)

[26] 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)

[27] Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)

[28] Saini, M.K., Gadde, R., Yan, S., Ooi, W.T.: Movimash: online mobile video mashup. In: Proceedings of the 20th ACM international conference on Multimedia. pp. 139–148 (2012)

[29] Shensa, M.J., et al.: The discrete wavelet transform: wedding the a trous and mallat algorithms. IEEE Transactions on signal processing **40**(10), 2464–2482 (1992)

[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. IEEE (2017)

[31] Thaiyalnayaki, S., Sasikala, J., Ponraj, R.: Detecting near-duplicate images using segmented minhash algorithm. International Journal of Advanced Intelligence Paradigms **12**(1-2), 192–206 (2019)

[32] Tuite, K., Snavely, N., Hsiao, D.y., Tabing, N., Popovic, Z.: Photocity: training experts at large-scale image acquisition through a competitive game. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 1383–1392 (2011)

[33] Uddin, M.Y.S., Wang, H., Saremi, F., Qi, G.J., Abdelzaher, T., Huang, T.: Photonet: a similarity-aware picture delivery service for situation awareness. In: 2011 IEEE 32nd Real-Time Systems Symposium. pp. 317–326. IEEE (2011)

[34] White, J., Thompson, C., Turner, H., Dougherty, B., Schmidt, D.C.: Wreckwatch: Automatic traffic accident detection and notification with smartphones. Mobile Networks and Applications **16**(3), 285–303 (2011)

[35] Wu, Y., Mei, T., Xu, Y.Q., Yu, N., Li, S.: Movieup: Automatic mobile video mashup. IEEE Transactions on Circuits and Systems for Video Technology **25**(12), 1941–1954 (2015)

[36] Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3733–3742 (2018)

[37] Yusof, N., Majid, N., Ismail, A.: Framework deduplication image detection assisted multimedia system using multi-technique. In: Proceedings of 2016 6th International Workshop on Computer Science and Engineering. pp. 402–406 (2016)
[38] Zauner, C.: Implementation and benchmarking of perceptual image hash functions (2010)