Cloud-Edge-Device Collaborative Image Retrieval and Recognition for Mobile Web

Yakun Huang[1(✉)], Wenwei Li[1], Shouyi Wu[1], Xiuquan Qiao[1], Meng Guo[2], Hongshun He[2], and Yang Li[2]

[1] Beijing University of Posts and Telecommunications, Beijing 100876, China
{ykhuang,wwli,sywu,qiaoxq}@bupt.edu.cn
[2] China Mobile Communications Research Institute, Beijing 100053, China
{guomeng,hehongshun,liyangyw}@chinamobie.com

Abstract. Efficient image retrieval and recognition are pivotal for optimal mobile web vision services. Traditional web-based solutions offer limited accuracy, high overhead, and struggle with vast image volumes. Transferring images for real-time cloud recognition demands stable communication, and large-scale concurrent requests strain computational and network resources. This paper introduces a distributed recognition approach, leveraging cloud-edge-device collaboration through edge computing's low latency and high bandwidth. We present a lightweight image saliency detection model tailored for mobile web, enhancing initial image feature extraction. Additionally, we introduce an edge-based, deep learning-driven method to amplify image retrieval speed and precision. We incorporate a location and popularity-based caching system to alleviate strains on cloud resources and network bandwidth during extensive image requests. Our real-world tests validate our approach: our saliency detection model outpaces the benchmark by reducing the model size by up to 94%, making it suitable for mobile web deployment. The proposed method improves retrieval accuracy by 40% over cloud-based counterparts and cuts response latency by over 60%.

Keywords: Cross-platform · Edge computing · Image retrieval

1 Introduction

Web browsers remain a primary gateway for mobile internet applications, facilitating seamless access to cross-platform services via browsers and integrated apps. With the progress in artificial intelligence and computer vision, image search and recognition have become crucial for leading mobile services, notably in browser-driven augmented reality (AR) and mini-programs [1,2]. These services often encompass tasks such as capturing frames, image preprocessing, detecting feature points, and matching against image databases. Hence, proficient and precise image recognition is essential for mobile web visual services.

Currently, there are two dominant techniques for image recognition in mobile web browsers: **(1) Browser-Based Image Recognition:** Utilizing frontend

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 562, pp. 474–494, 2024.
https://doi.org/10.1007/978-3-031-54528-3_26

frameworks like JSFeat, this approach leverages JavaScript computation libraries for image matching via algorithms such as SIFT [3], ORB [4], and AKAZE [5]. Though these conventional algorithms ensure robustness and accuracy, they are sluggish in matching speeds. The computational capacity of mobile web platforms is constrained, posing difficulties for extensive image recognition. **(2) Cloud-Based Image Recognition:** Here, resource-intensive tasks, including feature point detection and matching, are transferred to the cloud, accommodating large-scale mobile visual recognition demands [6]. Cloud servers adeptly enhance mobile web platforms' computational prowess. However, regular data and model exchanges introduce considerable communication latency. High-resolution media, such as images, audio, or video, under inconsistent communication conditions disrupts real-time visual computation and portrayal. Also, the cloud grapples with increased resource and bandwidth usage during extensive simultaneous requests, restricting the broad applicability of the mobile web.

This paper proposes a cloud-edge cooperative architecture for distributed mobile web image retrieval and recognition. This approach comprises a saliency detection-based image preprocessing module and an edge-conscious image retrieval acceleration. Developing this efficient method presents two challenges:

- **Image Resolution Increase.** Improved terminal camera performance leads to higher image resolutions. As image resolution increases, the time required for feature extraction using the same algorithm grows longer. Direct compression of real-time image frames can reduce feature extraction and retrieval accuracy. To address this challenge, we introduce a lightweight image saliency detection module tailored for lightweight mobile web platforms. This algorithm effectively localizes and segments the main subject in images. Notably, in mobile web AR applications, it efficiently extracts regions of interest from high-resolution images, reducing image frame transmission, accelerating subsequent feature point extraction and matching computations, and thus enhancing image recognition efficiency.
- **Traditional Image Matching Limitations.** Conventional image-matching algorithms with handcrafted local features offer robustness against viewpoint and lighting variations but necessitate geometric verification. As the volume of images in a template library grows, real-time recognition becomes challenging. Our study introduces an edge-aware retrieval acceleration module that enhances edge server image retrieval efficiency and optimizes computational resources. We combine deep learning-driven image retrieval with traditional matching methods, using deep learning to extract feature vectors and prescreen large image datasets, thus improving retrieval efficacy. Furthermore, we propose an image caching strategy based on geography and popularity, facilitating cloud collaboration. This strategy alleviates cloud computational and bandwidth demands, reduces mobile web latency, and bolsters edge server resource utilization, allowing for faster, more accurate large-scale image recognition.

To further validate the proposed cloud-edge collaborative distributed mobile web image retrieval and recognition method, we developed a prototype system

for large-scale image retrieval and recognition based on the KubeEdge framework. We conducted experiments and analyses in actual business scenarios. The primary contributions of this paper are:

- We designed a cloud-edge-device collaborative architecture to enhance the efficiency of large-scale concurrent web retrieval and recognition requests, especially boosting the efficiency of large-scale concurrent web retrieval and recognition requests.
- We proposed extracting the main area from real-time images to accelerate subsequent feature extraction speed and improve image recognition accuracy.
- We combined deep learning-based image retrieval with matching algorithms to further boost the accuracy and efficiency of mobile web image recognition requests.
- We designed a caching mechanism based on geographical location and popularity, reducing the pressure on the cloud layer and enhancing the speed and accuracy of recognition.

2 Design of Proposed Collaborative Architecture

2.1 Problem Description

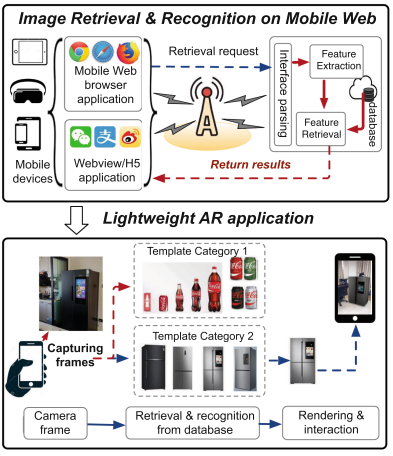

Fig. 1. Schematic of the mobile web image retrieval and recognition

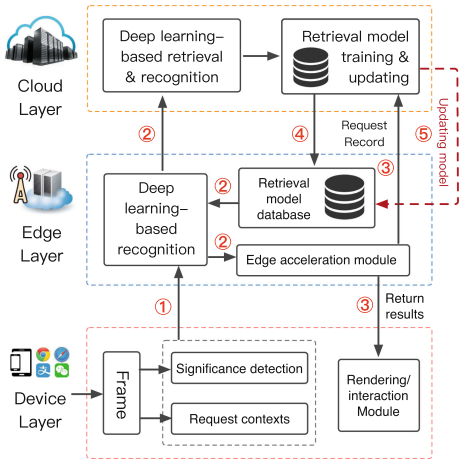

Fig. 2. Proposed architecture of mobile web image retrieval and recognition

Figure 1 depicts the sequence of steps taken when a mobile web application (encompassing browser apps and WebView/H5-based applications) sends an image retrieval and recognition request to a cloud server. Once the cloud server receives this request, it deciphers the API interface, proceeds with image feature extraction, and juxtaposes these features with its image database. After

performing the match and search, it arrives at the definitive retrieval and recognition outcome.

For a clearer perspective, let's consider the example of cross-platform mobile web AR recognition: (a) **Image Capture:** Through a browser, a user snaps a camera frame. (b) **Request Initiation:** The captured frame then sets off a retrieval and recognition petition to the cloud. (c) **Image Retrieval:** The cloud server contrasts the frame with its template library. (d) **Recognition & Rendering:** Upon finding a match, a 3D model tied to the identified object is displayed on the user's device. (e) **User Interaction:** This rendering culminates the user interaction, serving content or a service pertinent to the identified object. At its core, the primary aim of image retrieval and recognition activities is to methodically and promptly locate and align a target image from a designated image library, paving the way for subsequent operations and interactions. In contrast, the mission of image object recognition endeavors primarily hinges on ascertaining the category of the present object. The efficiency of retrieval and recognition is notably impacted by the size of the image library and the variations in features among targets of the same category. This paper zeroes in on the crucial challenge of streamlining precise image retrieval and recognition within resource-limited and cross-platform web applications, especially as it relates to immersive AR experiences and mobile multimedia visual services.

2.2 Overview of System Architecture Design

As depicted in Fig. 2, our proposed system utilizes a distributed computational structure spanning the mobile device, edge cloud, and remote cloud layers. Specifically, the device layer employs a saliency detection module to minimize data transmission, boosting the retrieval efficiency of both edge and remote clouds. Furthermore, richer context from the request enhances retrieval by providing personalized user data. Edge layer incorporates a deep learning-based image retrieval module, and an edge acceleration recognition module and synchronizes with the remote cloud's model library to improve retrieval accuracy. The acceleration module applies advanced techniques tailored to the user context. Emulating the edge layer's functionalities, the remote cloud layer houses a deep learning retrieval module, offering precise image results. Crucially, it hosts offline training for these models. Based on user request patterns, updated models are distributed to edge servers, optimizing efficiency and service quality.

For system edge node collaboration, we introduce a streamlined KubeEdge framework. It segments cluster operations for edge computing into two distinct components: Cloud Core (remote cloud) and Edge Core (edge side). The operational process for mobile web image retrieval and recognition unfolds as follows: **STEP 1**: Users activate the camera interface in browser apps to capture real-time frames. Using a saliency detection model, the primary subject is extracted from the image. This cropped image, together with the relevant context, triggers a retrieval request to the edge cloud. **STEP 2**: The edge cloud server processes the request, employing its deep learning recognition module to search its cached image library. If a match is found, results are sent back. If not, the task moves

to the remote cloud server. A successful remote retrieval results in the corresponding image being cached in the edge library. **STEP 3**: The edge server conveys the recognized results to the mobile web. This specific request record is saved and periodically synced with the remote cloud. **STEP 4**: Cloud servers, equipped with robust computational and GPU resources, periodically assimilate user-uploaded query images into training sets. This data assists in training deep learning image retrieval models. **STEP 5**: Based on real-time needs, edge servers regularly fetch updated models from the remote cloud, refining the model's adaptability and enhancing retrieval accuracy.

3 Methods of Proposed Collaborative Architecture

3.1 Image Preprocessing Based on Saliency Detection

The efficiency of image retrieval and recognition, based on frames captured by mobile devices and sent to servers, is predominantly affected by: (1) High-

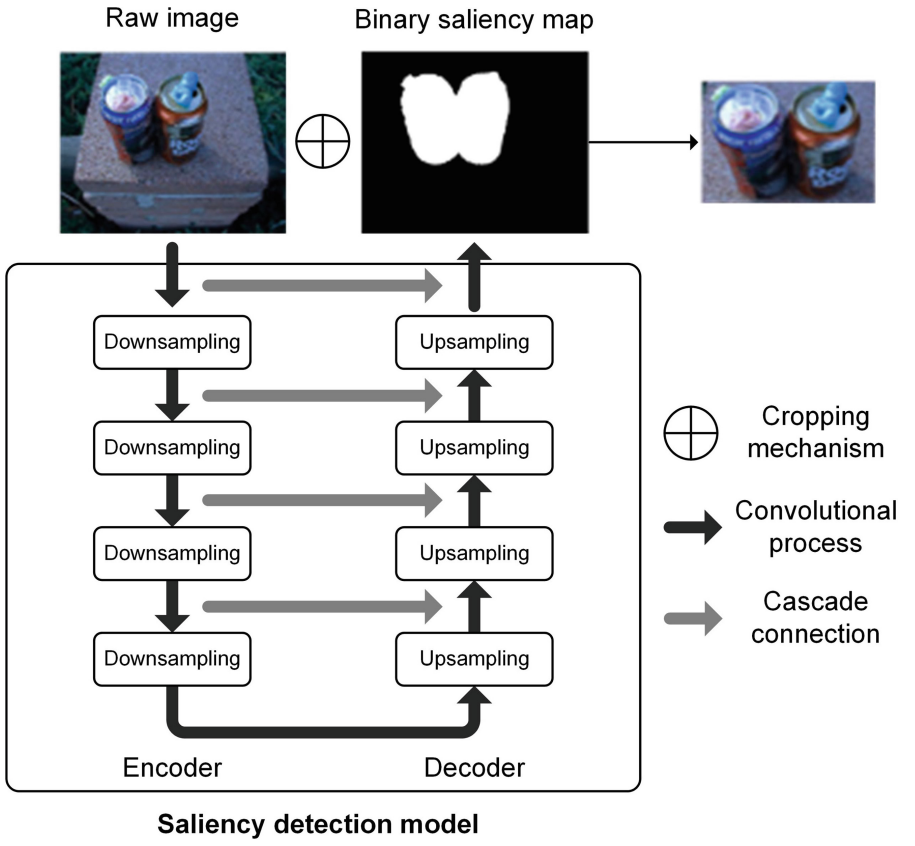

Fig. 3. Image preprocessing based on saliency detection

resolution Frames: Such frames necessitate a consistent, high-speed network, inducing considerable transmission lags. (2) Complex Backgrounds: Camera-captured keyframes often have a cluttered backdrop, impeding retrieval accuracy and speed. Employing deep learning for target segmentation intensifies computational demands, exacerbating retrieval and recognition delays.

Addressing the aforementioned challenges, we present a nimble saliency detection model optimized for widespread mobile web platforms. This model refines images captured by the camera, bolstering the accuracy and speed of server-side retrieval and recognition. Its core function is to identify, trim, and forward the primary subject portion of the image for server-side operations. As depicted in Fig. 3, the image preprocessing module, anchored on saliency detection, functions as follows: (1) The camera-captured raw image is inputted into the saliency detection model. (2) Using the derived binary saliency map, a region-centric cropping mechanism extracts the image's central subject. (3) Substituting the initial image, this refined version is relayed to the server, cutting down on data transmission and amplifying the efficiency and accuracy of the image retrieval and recognition.

Lightweight Saliency Detection. Fig. 4 delineates a saliency detection model tailored for the mobile web, addressing its inherent computational constraints. This model is structured to refine the clarity of saliency maps, featuring: (a) Foundation on Proven Architectures. Using the HED (Holistically-Nested Edge Detection) blueprint [7], the model augments the saliency map's resolution. The convolutional tiers of ShuffleNet V2 [8] lay the foundational network. (b) Multi-Scale Context Module (MSCM): It outputs a comprehensive saliency forecast. (c) Resolution Augmentation. Recognizing the global saliency map is merely 1/32 the resolution of the original, enhancing measures are adopted. Specifically, branches are integrated post the culminating layers of particular modules in ShuffleNet v2: Conv1, MaxPool, Stage 2, and Stage 3. These branches

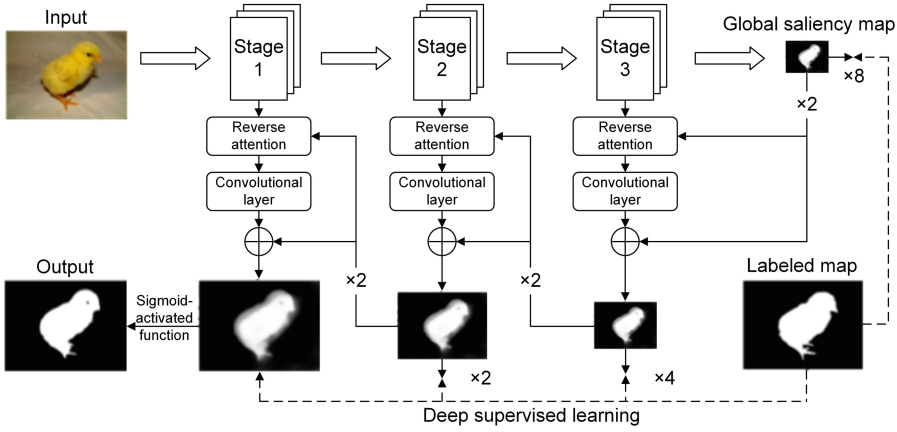

Fig. 4. Network structure of the saliency detection model

learn residual nuances at varying resolutions. Sequentially, features discerned from each layer aid in saliency map refinement. Final Projection: The superficial branch result from Stage 1 steers into a Sigmoid tier, culminating in the final predictive outcome. This design strikes a balance between elevating saliency map quality and ensuring a resource-conscious, nimble model apt for the mobile web.

Using ShuffleNet V2 in the backbone structure reduces the RAS [9] model size (initially based on VGG16) from 81MB to 3.5MB, but with a notable accuracy decrease. To counter this, we integrated a feature pyramid into ShuffleNet V2, including parallel 3×3, 5×5, and 7×7 depthwise separable convolutions. This efficiently processes spatial data from various target scales, improving accuracy. Figure 5 compares this revised structure to the original ShuffleNet, where "DWConv" denotes the depthwise convolution, splitting standard convolutions into channel-wise and point-wise tasks.

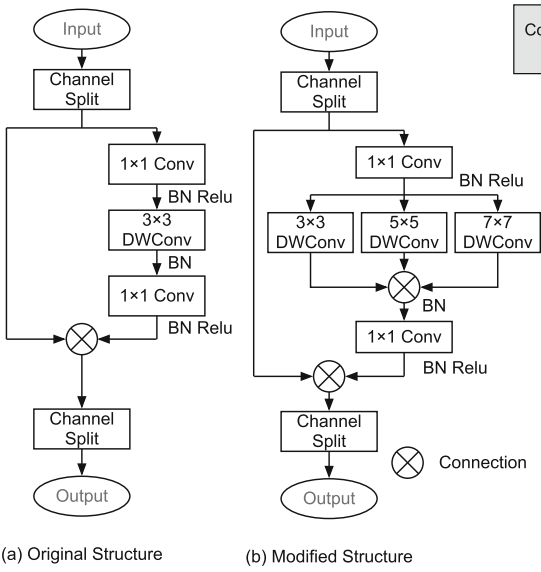

Fig. 5. Structure of the feature pyramid

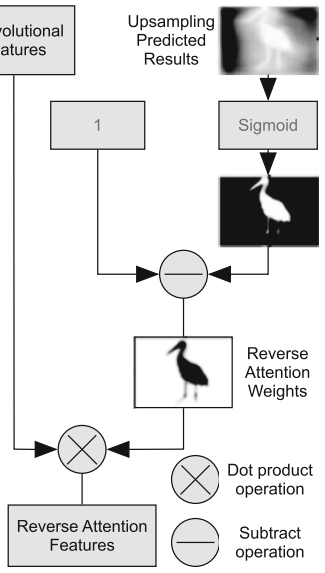

Fig. 6. Reverse attention module

Furthermore, in terms of network design, deep structures capture abundant semantic information but often miss spatial details, whereas shallow networks adeptly capture these nuances. The proposed model merges the strengths of both by using a branched learning approach. It adopts a top-down multi-branch residual learning strategy, integrating predictions with ground truth across scales through bilinear interpolation-based upsampling and residual units. This approach efficiently trains with fewer parameters, adaptively corrects errors at higher resolutions, and produces a more streamlined model. The computation for residual units is denoted in Eq. (1) and Eq. (2). Here, R_i symbolizes the current

branch's residual feature, and B_{i+1} represents the subsequent branch's saliency feature map. By upscaling and summing R_{i+1} and B_{i+1} by two, we obtain the saliency feature map B_i for the current branch, with G signifying the ground truth. Each branch i outputs a feature map upsampled by 2^i, which is then matched with the ground truth, effectuating deep supervised learning.

$$\{B_{i+1}\}^{up\times 2^{i+1}} \approx G \tag{1}$$

$$\left\{B_{i+1}^{up} + R_i\right\}^{up\times 2^1} = \{B_i\}^{up\times 2^i} \approx G \tag{2}$$

The branched structure incorporates a reverse attention module [9] tailored to discern details at diminished resolutions, thereby facilitating more accurate high-resolution predictions. As depicted in Fig. 6, this module generates reverse attention weights from the branch's prediction results, which are then multiplied by the original convolutional features, producing reverse attention features. This mechanism steers the multi-branch network's top-down learning, incrementally refining missing areas in saliency prediction to produce a map enriched in spatial detail.

As described in Eq. (3) and Eq. (4), A denotes the reverse attention weight, while T signifies the lateral output feature. The spatial position in the feature map is indicated by z, and the feature dimension is marked by c. The output feature F emerges from the point-wise multiplication of the original lateral output feature and the reverse attention weight. During this process, the prediction result from the $(i+1)^{\text{th}}$ branch lateral output is upsampled, then deducted from the Sigmoid-activated function of the upsampled output minus 1, forming the reverse attention weight A_i.

$$F_{z,c} = A_z \cdot T_{z,c} \tag{3}$$

$$A_i = 1 - Sigmoid(B_{i+1}^{up}) \tag{4}$$

Model Loss Function. For efficient training of the lightweight saliency detection model shown in Fig. 3, deep supervision is essential for each branch network. This research considers the output layer of each branch as a pixel-level binary classifier, utilizing the cross-entropy loss function to compute the loss for the i^{th} branch. The formula is:

$$\begin{aligned} &\ell_{\text{side}}^{(m)}\left(I, G, W, w^{(m)}\right) \\ &= -\sum_{x=1}^{|I|} G(z)\log \Pr\left(G(z)=1 \mid I(z); W, w^{(m)}\right) \\ &\quad + (1-G(z))\log \Pr\left(G(z)=0 \mid I(z); W, w^{(m)}\right) \end{aligned} \tag{5}$$

Here, $Pr(z)$ denotes the probability of pixel activation at position 'z' in the i^{th} branch. W encompasses parameters across all network layers, with i and G representing input and ground truth for each branch. The total loss function in

equation (4) aggregates losses from individual branches and the global saliency map, where M indicates the number of sub-loss functions.

$$c_x = \frac{M_{10}}{M_{00}}, c_y = \frac{M_{01}}{M_{00}} \tag{6}$$

Clipping Mechanism Based on Region Generation. Following the application of the saliency detection model, the original image produces a binary saliency map. Using the region generation mechanism [10], the cropping region's coordinates are determined. From this map, we derive the centroid coordinates (c_x, c_y), calculated as follows:

$$\mathrm{L}_{\text{side}}\left(I, G, W, w\right) = \sum_{m=1}^{M} \ell_{\text{side}}^{(m)}\left(I, G, W, w^{(m)}\right) \tag{7}$$

Subsequently, the standard deviation of the centroid coordinates is as follows:

$$\sigma_x = \sqrt{\frac{M_{20}}{M_{00}} - c_x^2}, \sigma_y = \sqrt{\frac{M_{02}}{M_{00}} - c_y^2} \tag{8}$$

wherein, the calculation methods for $M_{00}, M_{10}, M_{01}, M_{20}, M_{02}$ are as follows:

$$\begin{aligned} M_{00} &= \sum_{i,j} S_{i,j}, M_{10} = \sum_{i,j} i \cdot S_{i,j}, M_{01} = \sum_{i,j} j \cdot S_{i,j} \\ M_{20} &= \sum_{i,j} i^2 \cdot S_{i,j}, M_{02} = \sum_{i,j} j^2 \cdot S_{i,j}. \end{aligned} \tag{9}$$

Following this, the primary subject area is defined by the top-left and bottom-right corner coordinates, calculated using the following:

$$\begin{aligned} (x_1, y_1) &= (c_x + \alpha\sigma_x, c_y + \alpha\sigma_y), \\ (x_2, y_2) &= (c_x - \alpha\sigma_x, c_y - \alpha\sigma_y). \end{aligned} \tag{10}$$

Here, α is a hyperparameter that controls the comprehensiveness of the cropped primary subject. We set α to 0.3, ensuring the inclusion of the majority of the main subject while limiting background noise.

3.2 Edge Acceleration Recognition Module

Deep Learning-Based Image Recognition Acceleration. The image recognition module is pivotal for servers in image retrieval and identification. Dominant methods employ algorithms like SIFT, AKAZE, and ORB to extract image features and determine recognition based on matching points. Yet, with expanding image databases, these algorithms falter in real-time identification on mobile platforms. Recently, deep learning models for feature vector extraction and similarity calculations have demonstrated rapidity in extensive image

databases. Table 1 juxtaposes the speed of SIFT-based image matching with that of the ResNet deep learning model [11], both evaluated under consistent conditions using images of 500×500 dimensions across varying library sizes.

In the image recognition sequence, features are first extracted from the uploaded image and then matched with features from the image library. The table reveals that as the library size grows, the time disparity between traditional and deep learning feature-matching algorithms increases, underscoring the inadequacy of traditional algorithms for real-time retrieval on constrained mobile platforms. For optimal efficiency in large-scale image retrieval and recognition, we propose a deep learning-driven image retrieval acceleration technique comprising two essential phases. The initial phase involves computing the similarity of image feature vectors, swiftly procuring the top-K analogous images from a vast image library in relation to the input image. In the subsequent phase, feature points from these K images and the input image are derived using the SIFT methodology. Post geometric verification, the final results are ascertained by the count of matched points, ensuring accurate recognition. Notably, SIFT has garnered traction across diverse visual tasks for its dual strengths: efficiency and precision.

Table 1. Comparison of SIFT with deep learning-based features

	Number of Images	SIFT-Based	ResNet-Based
Feature Extraction	1	0.0880	0.1440
Feature Matching	1	0.0126	0.0001
	50	0.6371	0.0005
	500	6.3270	0.0012
	1000	12.5413	0.0018

Figure 7 elucidates this deep learning-centric acceleration method. The preliminary module, designated for image retrieval, begins by crafting a training dataset tailored to the present recognition context, succeeded by data refinement. Following this, a deep learning classification task, with ResNet as its core network, trains the model. The feature vector for the input image is sourced from the Softmax layer's input tensor within this network, leading to the extraction of a 2048-dimensional vector for every image. In an offline environment, feature vectors from the entire template image database (Gallery) are derived and cataloged in a matrix within a dictionary. In real-time usage, when a query arises, features from only the query image are extracted. Via matrix operations, distances between the query's feature vectors and those of the template library are gauged. Images are then sequenced based on these distances, with the top-K most similar images designated as the primary retrieval set.

The assessment of similarity among distinct images often leverages measures like the Euclidean distance and cosine similarity to gauge vector resemblances.

The Euclidean distance quantifies the length between two vector endpoints, as illustrated in Eq. (11). Conversely, cosine similarity represents the cosine value of the angle separating two vectors. Owing to its appreciable computational speed, outpacing that of the Euclidean distance, this paper opts for cosine similarity as its benchmarking metric. The methodology for computing this similarity is elucidated in the ensuing equation:

$$d(a,b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{11}$$

$$Sim = \frac{\sum_{i=1}^{n} (a_i \times b_i)}{\sqrt{\sum_{i=1}^{n} (a_i)^2} \times \sqrt{\sum_{i=1}^{n} (b_i)^2}} \tag{12}$$

To enhance the recall speed of the pre-selection module, this paper adds a network layer during training to condense the dimensionality of image feature vectors, improving computational efficiency and recognition speed. Furthermore, we

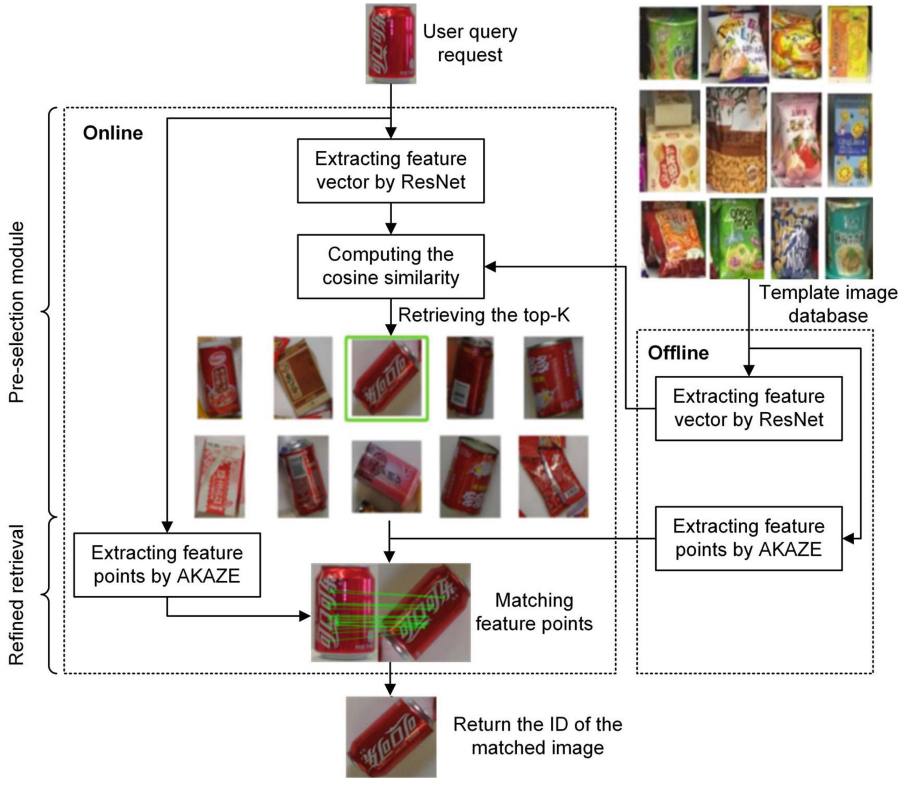

Fig. 7. Deep learning-based image recognition acceleration

employ the Hierarchical Navigable Small World (HNSW) algorithm for approximate nearest-neighbor searches. By indexing large-scale image feature vectors and prioritizing the most similar images, our approach significantly quickens image retrieval while maintaining search precision.

For the refined retrieval of the pre-selected image set, AKAZE features are utilized. These features, derived from the Hessian determinant's local maxima across scales, utilize binary descriptors to lessen computation and optimize real-time performance [12]. This paper incorporates an image recognition module on the cloud server, paralleling the edge server's capabilities. In real-world scenarios, the cloud server continuously receives user query images. With these fresh datasets, it refines and updates deep learning models, periodically syncing model parameters on both the cloud and edge, enhancing the adaptability of the deployed models.

Retrieval Caching Mechanism Based on Location and Popularity. Edge servers vary in service offerings based on user locations and application contexts. For instance, in museums, users predominantly request historical artifacts and renowned paintings, whereas in supermarket settings see frequent requests for everyday items. This suggests that template image retrieval frequency in databases correlates with both user location and image popularity. Consequently, the likelihood of querying the same template image can differ substantially across locations, resulting in varying access rates for identical data objects.

Given edge nodes' constrained cache capacity, formulating an adept cache replacement tactic is paramount when cache limits are met. Classical replacement strategies like Least Recently Used (LRU), First In First Out (FIFO), and Least Frequently Used (LFU) often underperform in terms of hit rates in dynamic contexts. Addressing this, our study proposes a cache replacement scheme anchored in content popularity. Specifically, we gauge the popularity of user-demanded content via its recent access frequency. Let $f_i(k)$ represent the instances content k is requested from node i within time T. Content k's access frequency, $F_i(k)$, is articulated in Eq. (13), where $\sum_{j=1}^{k} f_{i,j}$ tallies all content requests on node i during time T.

$$F_i(k) = \frac{f_i(k)}{\sum_{j=1}^{k} f_{i,j}} \tag{13}$$

Furthermore, the recency of a data request also influences the popularity calculation. If $T(k)$ denotes the time elapsed since the last request for content k, a smaller $T(k)$ indicates higher popularity. Considering the factors mentioned above, the popularity $P(k)$ of a cached data object in edge node i is defined as Eq. (14).

$$P(k) = \mu F_i(k) + (1 - \mu)\frac{1}{T(k)} \tag{14}$$

$\mu \in (0, 1)$ acts as a modulation factor, dictating the balance between recent access time and access frequency. Initially set at 0.5, μ might necessitate adjustments aligned with real-world use. When cache data on an edge node hits its

limit, data items with smaller $P(k)$ values are prioritized for replacement based on popularity. Users' geographical positions are typically sourced from their mobile devices' location systems, directing them to the closest edge node. However, in real scenarios, privacy-conscious users might opt out of location sharing. In these cases, a default method is employed: user queries are directly relayed to cloud servers for handling.

4 Experimental Analysis

4.1 Performance Analysis

Experimental Settings. Both cloud and edge service nodes utilize servers operating within Docker containers orchestrated by KubeEdge. Each server boasts an Intel®Xeon®Gold 5118 CPU @ 2.30GHz, 128 GB of RAM, and runs on Ubuntu 18.04.5 LTS. The system employs KubeEdge V1.3.0, Docker V19.03.12, and Mosquitto V1.6.12. Our infrastructure leveraged China Unicom's MEC service in Beijing, while cloud servers were based in Qingdao and Guizhou using Alibaba Cloud. Table 2 showcases transmission latency between server nodes for test requests originating from Beijing, with latencies averaged across 100 requests. It's clear that edge nodes markedly decrease transmission latency-a benefit that amplifies as the geographical gap between the cloud server and user widens.

Table 2. Experimental environment deployment

Access Node)	Location	Distance (Km)	Delay(ms)
Edge Node	Beijing	10	6
Cloud Node1	Qingdao	660	10
Cloud Node2	Guiyang	2083	78

Analysis of Experimental Results. In this section, we contrast the performance of our proposed methodology with cloud-based services within the aforementioned collaborative cloud environment. Table 2 presents a comparison of processing delay performance across different methods, considering varying data sizes. Notably, the edge server caches 500 images, while the cloud computing center retains the complete image dataset. During practical testing, user-initiated image retrieval requests are first addressed by the proximate edge node. If the recognition result meets expectations, it's promptly relayed to the user terminal. However, if the search query doesn't find a match in the edge node's cached image database, the request is rerouted to the cloud center. Subsequently, the cloud center's search outcome is dispatched to the device.

Table 3. Retrieval time comparison of different methods. Unit: Milliseconds (ms).

	Number of Images	Data Fetch	Model Loading	Image Retrieval	Image Matching	Processing Delay	Total Delay
Only Edge	500	24	157	0.4	185	6	372.4
Only Cloud	10000	25	178	7.3	183	78	471.3
	100000	23	200	53.1	191	78	545.1
Collaboration	10000	26	179	7.5	184	81	477.5
	100000	26	200	54.6	188	81	549

The key takeaways from our experimental results are: (1) For cloud computing centers with image databases of 10,000 and 100,000 images, relying solely on edge nodes for request handling yields reduced latencies compared to alternative techniques. Nonetheless, this approach isn't optimal for larger databases, positioning it as ideal for swift retrieval within more confined image sets. (2) Central cloud centers, while capable of accommodating expansive image searches, entail a longer processing delay, diminishing user satisfaction. (3) Our suggested cloud-edge collaboration technique effectively addresses a considerable chunk of user queries at the edge. By refining edge data caching tailored to specific business dynamics and user query trends, the majority of requests are catered to at the edge. This strategy retains its potency even amidst significant shifts in business demands, ensuring uninterrupted user services. Notably, for extensive image retrievals, the use of edge nodes significantly trims user request response durations.

4.2 Saliency Detection Analysis

After showcasing the efficacy of our cloud-edge collaboration in previous experiments, we now delve into evaluating the image preprocessing module's performance, which is rooted in saliency detection and implemented on the mobile web. This section outlines the datasets and experimental configurations, contrasts our method with standard benchmarks, and ends by gauging the module's performance on diverse mobile web platforms.

Datasets and Configuration. We quantitatively validate our model on three key datasets often used in saliency detection: **MSRA-B** [13]: A dataset of 5,000 images, divided into 3,000 for training and 2,000 for testing. **HKU-IS** [14]: A collection of 4,447 images. **ECSSD** [15]: It contains 1,000 images. Training of the saliency model was expedited using the NVIDIA TITAN XP GPU on the PyTorch framework with the Stochastic Gradient Descent (SGD) optimizer. The initial learning rate was set at 1×10^{-8}, with crucial parameters like batch size, momentum, and learning rate decay coefficient held constant. As training loss reached a certain level, the learning rate was judiciously reduced. To counteract overfitting, strategies such as horizontal flipping were integrated into our data augmentation, bypassing the use of a validation set.

Analysis of Experimental Results. We examined the efficiency of saliency detection models, comparing with methods like DRFI [16], DCL+ [17], DSS [18],

ASNet [19], PicaNet [20], and RAS [9] using the MSRA-B training set. Metrics analyzed encompassed model size, time overhead, F-Measure, and MAE, with F-Measure representing the harmonic mean of precision and recall. Figure 8 shows the overhead of different saliency detection methods on the web. Results reveal our proposed lightweight model is 3.5MB, marking a 95.7% reduction from the second-ranking RAS. In comparison, the ASNet model is 363MB, introducing notable latency for mobile web.

As shown in Fig. 9, we compared the average processing time of our method with benchmark saliency algorithms on the MSRA-B test dataset using 500×500 images. Our method improved inference efficiency by about 92.3%, 97.9%, and 65.6% over RAS, DRFI, and DCL+ methods, respectively. This indicates our lightweight model's suitability for real-time mobile web use, whereas traditional methods lag in performance. In saliency detection, this work introduces an innovative lightweight method optimized for mobile web platforms, transferring some computational tasks from the cloud, thereby boosting cloud computing center throughput.

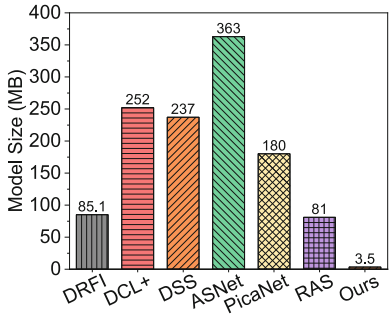

Fig. 8. Analysis of model sizes of several saliency detection algorithms

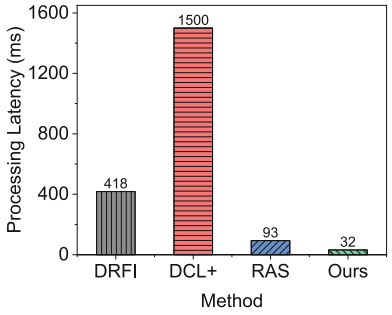

Fig. 9. Execution time of different saliency detection algorithms

Experiments on the MSRA-B, HKU-IS, and ECSSD datasets evaluated model accuracy. Figure 10 and Fig. 11 juxtapose the performance of the DRFI, DCL+, and RAS algorithms with our proposed lightweight saliency detection method using the F-Measure and MAE metrics. Ideally, a high F-Measure and low MAE are sought. The results reveal that our lightweight model boasts a 94% size reduction compared to the advanced RAS model and reduces inference latency by 62%. The inclusion of a pyramid structure further curtails accuracy loss. Notably, our method's F-Measure and MAE scores surpass those of leading methods, proving its efficiency and precision for practical use on mobile web.

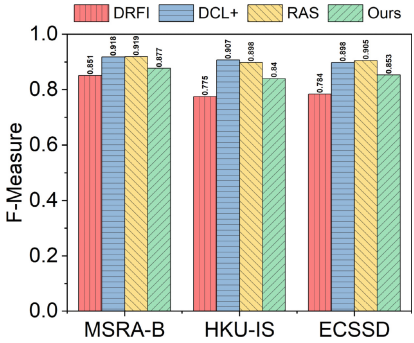

Fig. 10. F-Measure performance

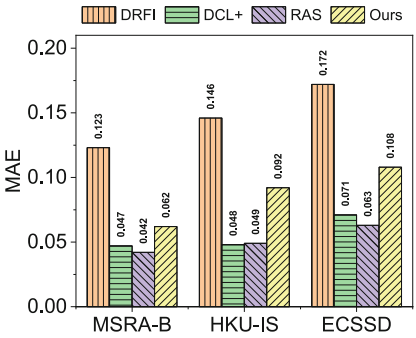

Fig. 11. MAE performance

For deployment and performance, the crafted lightweight saliency detection model is readily deployable and executable on common mobile web browsers. During implementation, we transitioned the model from Keras's h5 format to a mobile web-friendly JSON format using the TensorFlow.js Converter. Table 3 showcases the inference durations and frame rates when using our model on diverse mobile devices, with test images at a 500×500 resolution in the Chrome browser. All results reflect the average of ten independent tests.

The results suggest that the mobile device's performance significantly impacts the model's inference time and frame rate. For example, a more basic device like the Xiaomi6 achieves about 5 frames per second. In contrast, the Samsung Galaxy Note20 reaches roughly 10 frames per second, and the iPhone 13 Pro hits around 15 frames per second. While there's a performance variation across devices with diverse specifications, the model ensures real-time image processing even on less advanced devices, handling multiple frame inferences within a second.

Table 4. Test on different mobile devices.

Device (CPU)	Image Size	Running Time (ms)	Frame Rate
Xiaomi6	500×500	180–200	5–5.5
Honor p20 pro	500×500	122–149	6.7–8.2
Galaxy Note20	500×500	95–108	9.3–10.5
iPhone X	500×500	103–120	8.3–9.7
iPhone 11	500×500	96–109	9.2–10.4
iPhone 13 Pro	500×500	63–79	12.6–15.9

4.3 Analysis of Image Retrieval Acceleration

Deep Learning-Based Image Recognition. To assess the effectiveness and efficiency of the deep learning-based image recognition method, we employed

the Retail Product Checkout dataset [21] from Megvii, consisting of images for 200 distinct merchandise items. We partitioned it into a template library with 600 images (three per category) and a query library of 8,397 images taken from various perspectives. We further evaluated the benefits of the terminal saliency detection module by using a lightweight model to focus on the main portions of the query images. Unneeded backgrounds were removed through a cropping process, after which we compared both the original and processed images.

Using the given dataset, this study assesses the saliency detection model using two metrics: "Recall/Recognition Accuracy" and "Image Recognition Time." Fig. 12 shows the outcomes for images processed both before and after the application of the lightweight saliency detection model. For image retrieval, cosine distances are calculated from the image feature vectors, which are then ranked to pinpoint the most similar images. The model's efficacy is gauged using TOP-K recall accuracy, as outlined in the following formula.

$$Accuracy_{recall} = \left(\sum_{i=1}^{N} y_i \right) /N. \tag{15}$$

Let N denote the number of images in the test set. The indicator variable y_i is set to 1 if at least one of the top k images retrieved by the model matches the query image's category; otherwise, it's 0. In the experiments, 8,397 query images were matched against a set of 600 template images.

Figure 12 highlights several key findings: (1) The saliency detection module effectively isolates the main subject in user images, removing extraneous background details. As a result, vital information dominates the feature vectors, leading to a marked increase in recall accuracy across Top-1, Top-5, Top-10, and Top-20 retrievals. The "retrieval + matching" approach involves refining the top-10 recalled images with a precise image-matching algorithm, with final identification based on the number of matching points. (2) Introducing the proposed image matching technique enhances recall accuracy beyond that of sole image recognition, evident in the Top-1 column. Post saliency detection, the image recognition accuracy rises to 0.8378, an over 40% surge from before and a significant jump from the original 0.2847. (3) After preprocessing, our method consistently outperforms in recall accuracy for all Top-K categories, highlighting the efficiency of the deep learning-based recognition approach. This translates to enhanced performance in mobile web search and recognition, thereby elevating user experience.

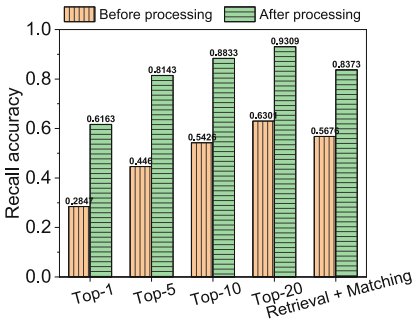

Fig. 12. Image recognition accuracy

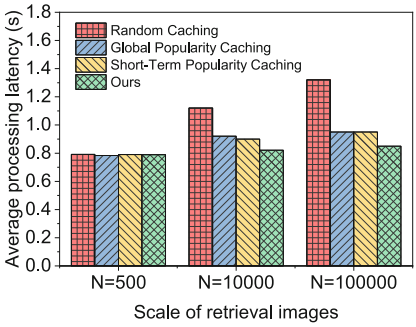

Fig. 13. Performance of different image retrieval cache strategies

Table 5 details the time overhead associated with image recognition computations on mobile Web platforms across various stages. Notably: (1) Saliency Detection Overhead: Using the lightweight saliency detection model on a mobile Web browser introduces an overhead of 180–200ms, increasing the preprocessing time from 10ms to 210ms. For single-image matching, the saliency detection module efficiently eliminates extraneous background, decreasing the feature point extraction and descriptor generation time using the AKAZE algorithm from 123ms to 51ms. The aggregate processing time for matching the top 10 images retrieved shows a marked difference in comparison to the preprocessed duration. (2) Image Retrieval Overhead: As images are standardized to a consistent size before feature vector extraction, the expected processing time stays relatively stable pre and post-preprocessing, with slight variances. The overall time for the image recognition service drops from 1352ms to 824ms, a 39% decrease in service response time. Such a refined strategy optimizes computational requirements, leading to an enhanced user experience on mobile Web platforms.

Table 5. Cost of image recognition on the mobile web. Unit: Milliseconds (ms).

	Terminal Preprocessing	Image Retrieval	Single Image Matching Time	Total Image Matching Time	Total Time
Pre-processing	10	112	123	1230	1352
Post-processing	210	104	51	510	824

Cache Strategy Analysis Based on Location and Popularity. This section assesses the efficacy of our caching strategy, which emphasizes geographical location and image popularity in real-world situations. We maintain consistent network topology, computational nodes, and experimental parameters as in prior experiments. Figure 13 examines the average request processing latency across various user request volumes, comparing multiple caching strategies: (1) Random Caching. Images are cached at edge nodes from the cloud randomly.

(2) Global Popularity Caching. Edge nodes cache images based on their overall request frequency and popularity. (3) Short-Term Popularity Caching. Edge nodes cache images that have recently shown high request frequency and popularity. (4) Proposed Method. This strategy considers both the user's location and image request popularity. The objective is to discern the impact of these strategies on latency and to highlight the benefits of our suggested method.

Key observations include: (1) Scale of Image Retrieval. At a retrieval scale of 500 images, the latency differences across methods are marginal. All images can be cached at edge nodes, resulting in comparable average latencies across strategies. (2) Elevated Retrieval Scale. As retrieval scale grows, our method, which merges geographical and short-term popularity considerations, primarily addresses user requests at edge caches, negating the need to contact the distant cloud. This delivers superior latency performance. (3) Popularity-Centric Strategies. Although popularity-focused methods (global, short-term, and our proposed) surpass random caching in latency, they might compromise retrieval rates for less-demanded images, necessitating more cloud interactions. Conclusively, our geography and popularity-centric caching method predominantly meets user retrieval needs, optimizing the average processing latency.

5 Conclusion

This paper discusses the challenges of mobile web image retrieval and recognition services, notably prolonged latency and decreased accuracy in real-world settings. We introduce a cloud-edge collaborative computing architecture that uses edge computing for timely request responses. Algorithms are deployed on both cloud and edge nodes via Docker containers, centrally managed from the cloud to reduce the strain on cloud-based services. We present a lightweight saliency detection model that increases computational efficiency by distinguishing the primary subject in user images, eliminating background noise, and facilitating feature point extraction with region-based cropping. Edge-based caching, designed around geographical location and popularity, maximizes the restricted cache capacity of edge nodes to elevate content accessibility. Integrating image retrieval with image matching, we compare feature vectors to quickly pinpoint the top-K images from a vast library, then use the AKAZE algorithm to extract feature points from a reduced candidate pool, finalizing matches based on point congruence. Tests reveal our approach improves accuracy by 47% and reduces response times by over 60% compared to standard methods.

Acknowledgment. This research was funded in part by the National Natural Science Foundation of China under Grant 62202065, in part by the Project funded by China Postdoctoral Science Foundation 2022TQ0047 and 2022M710465, and in part by Beijing University of Posts and Telecommunications-China Mobile Research Institute Joint Innovation Center.

References

1. Qiao, X., Ren, P., Dustdar, S., Liu, L., Ma, H., Chen, J.: Web ar: a promising future for mobile augmented reality-state of the art, challenges, and insights. Proc. IEEE **107**(4), 651–666 (2019)
2. Qiao, X., Ren, P., Nan, G., Liu, L., Dustdar, S., Chen, J.: Mobile web augmented reality in 5g and beyond: Challenges, opportunities, and future directions. China Commun. **16**(9), 141–154 (2019)
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**, 91–110 (2004)
4. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: an efficient alternative to sift or surf. In: 2011 International Conference on Computer Vision, pp. 2564–2571. IEEE (2011)
5. Alcantarilla, P.F., Solutions, T.: Fast explicit diffusion for accelerated features in nonlinear scale spaces. IEEE Trans. Patt. Anal. Mach. Intell **34**(7), 1281–1298 (2011)
6. Qiao, X., Ren, P., Dustdar, S., Chen, J.: A new era for web ar with mobile edge computing. IEEE Internet Comput. **22**(4), 46–55 (2018)
7. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1395–1403 (2015)
8. Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 122–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_8
9. Chen, S., Tan, X., Wang, B., Lu, H., Hu, X., Fu, Y.: Reverse attention-based residual network for salient object detection. IEEE Trans. Image Process. **29**, 3763–3776 (2020)
10. Lu, P., Zhang, H., Peng, X., Jin, X.: An end-to-end neural network for image cropping by learning composition from aesthetic photos. arXiv preprint arXiv:1907.01432 (2019)
11. Jian, S., Kaiming, H., Shaoqing, R., Xiangyu, Z.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision & Pattern Recognition, pp. 770–778 (2016)
12. Prakash, C.S., Panzade, P.P., Om, H., Maheshkar, S.: Detection of copy-move forgery using akaze and sift keypoint extraction. Multimedia Tools Appli. **78**, 23535–23558 (2019)
13. Liu, T., Yuan, Z., Sun, J., Wang, J., Zheng, N., Tang, X., Shum, H.Y.: Learning to detect a salient object. IEEE Trans. Pattern Anal. Mach. Intell. **33**(2), 353–367 (2010)
14. Li, G., Yu, Y.: Visual saliency detection based on multiscale deep cnn features. IEEE Trans. Image Process. **25**(11), 5012–5024 (2016)
15. Shi, J., Yan, Q., Xu, L., Jia, J.: Hierarchical image saliency detection on extended cssd. IEEE Trans. Pattern Anal. Mach. Intell. **38**(4), 717–729 (2015)
16. Jiang, H., Wang, J., Yuan, Z., Wu, Y., Zheng, N., Li, S.: Salient object detection: a discriminative regional feature integration approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2083–2090 (2013)
17. Li, G., Yu, Y.: Deep contrast learning for salient object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 478–487 (2016)

18. Hou, Q., Cheng, M.M., Hu, X., Borji, A., Tu, Z., Torr, P.H.: Deeply supervised salient object detection with short connections. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3203–3212 (2017)
19. Wang, W., Shen, J., Dong, X., Borji, A.: Salient object detection driven by fixation prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1711–1720 (2018)
20. Liu, N., Han, J., Yang, M.H.: Picanet: learning pixel-wise contextual attention for saliency detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3089–3098 (2018)
21. Wei, X.S., Cui, Q., Yang, L., Wang, P., Liu, L.: Rpc: a large-scale retail product checkout dataset. arXiv preprint arXiv:1901.07249 (2019)