Multimedia Tools and Applications (2024) 83:20243–20263
https://doi.org/10.1007/s11042-023-16164-5

Learning visual similarity for image retrieval with global descriptors and capsule networks

Duygu Durmuş[1] · Uğur Güdükbay[1] · Özgür Ulusoy[1]

Received: 26 July 2022 / Revised: 7 May 2023 / Accepted: 3 July 2023 /
Published online: 31 July 2023

Abstract
Finding matching images across large and unstructured datasets is vital in many computer vision applications. With the emergence of deep learning-based solutions, various visual tasks, such as image retrieval, have been successfully addressed. Learning visual similarity is crucial for image matching and retrieval tasks. Capsule Networks enable learning richer information that describes the object without losing the essential spatial relationship between the object and its parts. Besides, global descriptors are widely used for representing images. We propose a framework that combines the power of global descriptors and Capsule Networks by benefiting from the information of multiple views of images to enhance the image retrieval performance. The Spatial Grouping Enhance strategy, which enhances sub-features parallelly, and self-attention layers, which explore global dependencies within internal representations of images, are utilized to empower the image representations. The approach captures resemblances between similar images and differences between non-similar images using triplet loss and cost-sensitive regularized cross-entropy loss. The results are superior to the state-of-the-art approaches for the Stanford Online Products Database with Recall@K of 85.0, 94.4, 97.8, and 99.3, where K is 1, 10, 100, and 1000, respectively.

Keywords Deep learning · Neural networks · Capsule networks · Global descriptors · Image retrieval · Triplet loss · Cost-sensitive regularized cross-entropy loss

1 Introduction

Deep convolutional neural networks can be utilized for various tasks such as image classification, object detection, and image retrieval. An image retrieval system enables browsing and retrieving images relevant to the given query image from a large dataset of images. Searching and finding matching images across a large and unstructured image collection is a common problem in computer vision systems [31]. Image similarity and representation are significant for image retrieval. Thus, learning visual similarity is crucial, and visual

✉ Uğur Güdükbay
gudukbay@cs.bilkent.edu.tr

[1] Department of Computer Engineering, Bilkent University, Bilkent, Ankara 06800, Turkey

representation becomes critical to image matching and retrieval tasks due to adverse effects of angle, background clutter, or illumination. The best-performing approaches benefit from the local invariant features, such as scale-invariant feature transform (SIFT) [29], and strategies, such as bag-of-words (BoW) [41], and vector of locally aggregated descriptors (VLAD) [17].

In recent years, convolutional neural networks (CNNs) have become popular with the outperforming performance in computer vision tasks such as image retrieval [51]. In the latest works for image retrieval, based on the activations within CNNs, global descriptors generated by sum pooling of convolutions (SPoC) [3], generalized mean pooling (GeM) [34], attention-aware generalized mean pooling (AGeM) [11], maximum activation of convolutions (MAC) [47], have been used for computer vision tasks, especially for visual recognition [7]. Furthermore, several recent works benefit from using the output of the last fully connected layers as global image descriptors in the case of limited descriptor dimensionality [29]. Even if global descriptors' success varies by dataset, global descriptors lead to various image representations by activating different images' regions.

Hinton et al. [38] introduce an alternative neural network based on capsules and an algorithm for dynamic routing between capsules that learns to encode valuable information about the objects without losing the intrinsic spatial relationship between the parts and wholes. Unlike max-pooling used in CNNs, capsule networks do not discard information about the precise position of the parts of objects within the image [38]. It is because pooling operations confuse the information about the instantiation characteristics. The routing-by-agreement in capsule networks utilizes modules and capsules rather than max-pooling, while capsules replace scalar outputs in CNNs with vector outputs. Thus, capsule networks recognize the images invariant to viewpoint since this architecture can inherently learn higher dimensional pose configuration of the images [24]. Although capsule networks perform superior on the MNIST dataset, they do not perform well on complex datasets such as CIFAR-10. The reason is that the higher dimensionality of CIFAR-10 data requires more complex encodings of the images, where capsules become insufficient.

Since retrieving images relevant to a given query image from a large dataset of images is a difficult task, we aim to improve the image retrieval performance of convolutional neural networks by combining global descriptors and the concept of capsule networks using the information of multiple views of images to empower capsule networks with various global descriptors. To the best of our knowledge, this is the first proposed architecture that replaces primary capsules of capsule networks with multiple global descriptors. The CNN backbone of pre-trained SE-ResNet50 and specific mechanisms, including the Spatial Group-wise Enhance (SGE) module and self-attention layer, are included to strengthen the feature learning from the given input triplets. Instead of using a single network to learn the classification of individual images, a triplet design is considered to capture the resemblances between similar images and differences between non-similar images for given multiple views of image triplets.

The rest of the paper is organized as follows. Section 2 presents the background relevant to the proposed framework, including image retrieval, deep learning, convolutional neural networks, and related work. Section 3 describes the proposed network architecture, including the CNN backbone, Capsule, Descriptor, and Concat modules. Section 4 describes the network training process by utilizing online triplet loss, hard negative mining, and cost-sensitive regularized cross-entropy loss. Section 5 provides the implementation details, including the time and memory efficiency, feature extraction process, evaluation metrics, and the dataset. Section 6 discusses experimental results and provides a comparison to the-state-of-art methods. Finally, Section 7 concludes and discusses future work.

2 Related work

Many strategies are applied for image retrieval tasks in recent works, especially with the emergence of deep convolutional neural networks. Dai et al. [8] propose a method called batch feature erasing to optimize feature representation for image retrieval datasets that significantly improve general image retrieval benchmarks. Kim et al. [22] present an attention-based ensemble using multiple attention masks so that each learner can learn different parts of the object with a divergence loss encouraging diversity among learners. Jun et al. [18] introduce a new framework for image retrieval, called CGD, that utilizes multiple global descriptors to get an ensemble effect. Our proposed framework differs from [18] with using the Capsule module, how the Capsule module is concatenated with the Descriptor module, the summation of Cost-Sensitive Regularized Cross-Entropy Loss and Classification Loss for network training, and the Concat module architecture. Kayhan et al. [21] propose using textural properties of objects for content-based image retrieval based on a weighted combination of color and texture features. Please refer to Kapoor et al. [20] for a recent survey on content-based image retrieval techniques using deep learning.

Computer vision tasks, including image classification [26] and object detection [35], benefit from global image descriptors based on deep convolutional neural networks. Babenko et al. [3] suggest sum pooling of convolutions (SPoC) based on a sum-pooling aggregation descriptor, which improves the state-of-art retrieval accuracy with the combination of good design choices on standard retrieval datasets. In addition to the SPoC descriptor, recent works construct competitive visual representations from the activations of convolutional layers by utilizing maximum activation of convolutions (MAC), regional maximum activation of convolutions (R-MAC) [47], spatial max-pooling [1], cross-dimensional weighting (CroW) [19], AGeM [11].

Learning visual similarity is crucial for image retrieval, aiming to find similar images in a large dataset. Current works use loss functions in deep metric learning, such as contrastive loss [4, 31] and triplet loss [8, 18]. Furthermore, more advanced losses are designed, such as N-pair loss [42] and lifted structure loss [43]. Ge et al. [10] propose a novel hierarchical triplet loss (HTL) that encourages the model to learn more discriminative features, leading to faster convergence and better performance. Sun et al. [44] introduce circle loss on various deep feature learning tasks that achieve superior performance on image retrieval datasets. Hard sample mining methods, such as distance weighted sampling [22, 48], semi-hard, and hard negative mining [6, 39], are also crucial for image retrieval performance in addition to loss functions.

An alternative neural network named *capsule* network is introduced by Hinton et al. [38]. Hinton et al. [14] also introduce a new iterative routing procedure between capsule layers using the Expectation-Maximization (EM) algorithm. It allows the output of each lower-level capsule to be routed to a capsule in the layer above so that active capsules receive a cluster of similar pose votes. They describe a new type of capsule system in which each capsule has a logistic unit to represent the presence of an entity and a 4×4 pose matrix that could learn to represent the relationship between that entity and the pose. Since the objects can be described as a set of geometric organized parts, Kosiorek et al. [25] propose an unsupervised capsule autoencoder (SCAE). It explicitly uses geometric relationships between parts and, thus, is robust to viewpoint changes. Kinli et al. [24] propose two different triplet-based designs of capsule network architecture with different feature extraction methods for clothing retrieval tasks. Ozcan et al. [33] introduce Quaternion Capsule Networks (QCNs). Quaternions can be considered a regularization of the rotation representation for capsules, representing the pose information of capsules and their transformations, and requires fewer parameters than matrices. Ribeiro et al. [36] describe a new capsule routing algorithm based on Variational Bayes for learning a mixture of transforming Gaussians. For 3D point cloud construction, canonicalization, and unsupervised classification, Sun et al. [45] propose a self-supervised capsule architecture that learns an object-centric representation.

Capsule networks are not as effective as CNNs by themselves for complex datasets. Thus, as mentioned above, several approaches such as matrix capsules [14], Stacked Capsule Encoders [25], Quaternion Capsules [33], and Bayesian Routing [36] are combined with capsule network design. Considering this motivation and using multiple global descriptors as in [18], the idea is to combine this concept with robust global image descriptors to enhance image retrieval performance for complex datasets. Babenko et al. demonstrate that fine-tuning a pre-trained CNN with domain-specific data can improve retrieval performance on relevant datasets [2]. Therefore, our proposed framework benefits from a pre-trained CNN. The aim is to learn spatial relationships between object and object parts with capsule networks and capture richer visual information with multiple global descriptors. It is trained in an end-to-end manner without the computational cost of the ensemble technique.

3 Proposed framework

We propose a practical framework for the image retrieval task comprising a pre-trained CNN backbone and three modules. Each image in the image triplet is fed to this framework. Rather than building and training the CNN backbone from scratch, we prefer a pre-trained model as a starting point because a complex dataset including multiple views of images is used, and a pre-trained CNN is useful to accelerate the training process and distinguish objects due to the prior information related to the objects. The first module is a revised capsule network, named the Capsule module, used to learn the location and orientation of object parts relative to one another, enabling better recognition of object parts from various points of view. The second one, the Descriptor module, obtains various image representations by activating different images' regions and capturing differential features. The third module is the Concat module, which concatenates the outputs of the first and the second modules to obtain richer information about the given input image. The Concat module fine-tunes the CNN backbone with both modules to learn richer image representation. Therefore, our aim of learning spatial relationships between object and object parts with capsule networks and capturing richer visual information with multiple global descriptors is possible in an end-to-end manner without the computational cost of the ensemble technique.

3.1 CNN backbone

Although using a single convolutional layer is sufficient for simple datasets such as MNIST [27], a complex dataset requires more convolution layers to learn better feature embeddings. Thus, rather than using a single convolutional layer, the CNN backbone is utilized for obtaining feature embeddings. Rather than building and training the CNN backbone from scratch, we prefer a pre-trained model as a starting point because a complex dataset, including multiple views of images, is used. A pre-trained CNN is useful to accelerate the training process and distinguish objects due to the prior information related to the objects.

Due to memory restrictions, the proposed framework can benefit from pre-trained CNN backbones, including two well-known CNN architectures, ResNet [13] and VGG [40]. However, since ResNet provides better results than VGG in our experiments and for the similar dataset in [18], SE-ResNet-50 is selected after experiments on BN-Inception [16], ShuffleNet-v2 [30], ResNet-50 [13] and SE-ResNet-50 [15], SE-ResNet-50 is preferred as a CNN backbone. A strong baseline of ResNet, empowered with SE blocks, is called SE-ResNet50, and it is used as a CNN backbone as in [18], as shown in Fig. 1. This pre-trained model is trained on ImageNet data used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [37]. Since this neural network has already learned to classify images into 1000 object categories, a pre-trained network with learned weights about those object categories is helpful for the training process of the proposed architecture.

The CNN backbone is composed of four stages. We remove the down-sampling operation between Stages 3 and 4 as in [8, 18] to obtain richer image representations. After this elimination, we get a larger feature map of 14×14. First, batch normalization [16] is applied to the output of the pre-trained CNN backbone. After that, SGE mechanism [28] empowering the feature learning and suppressing the noise, and the self-attention layer relating multi-level dependencies across the image regions [50] are performed, respectively.

3.2 Capsule module

In the proposed framework, the Capsule module is a revised capsule network and capsule layers are the same as the original design [38], except for the primary capsules. Rather than using primary capsules, a concatenation of reshaped global descriptors is fed to the class capsule layer. The purpose of primary capsules is to capture the sub-features into capsules. However, since there are 10,517 product classes in the training dataset (after the validation dataset is removed from the training dataset), it is difficult to capture all features in each product class. It requires many capsules that lead to high computational costs. Therefore, instead of using a primary capsule layer, the branches of the Descriptor module are l_2 normalized, concatenated, and reshaped to be fed into the class capsule layer, as the Descriptor module significantly includes various pooling functions that highlight image information. The pooling function improves the model's performance for image retrieval tasks [3, 34, 47].

There are twelve 16-dimensional capsules in class capsule layers. The number of classes is 10,517 for the training dataset. Therefore, the number of superclass categories is chosen for the number of channels as 12, which can be seen in Fig. 1. Activations and the latent capsule vectors of the class capsule layer are calculated using dynamic routing with three iterations. We remove the reconstruction method and designed loss for capsule networks [38] in the proposed framework. A self-attention layer follows the class capsule layer. The output embedding is fed into the Concat module.

3.3 Descriptor module

Based on the experiments for twelve configurations of SPoC [3], GeM [34], and MaC [47] in [18] for a similar dataset, the most effective configuration of global descriptors is the combination of SPoC and GeM. Therefore, using the same motivation, the Descriptor module consists of three branches, each with one global descriptor.

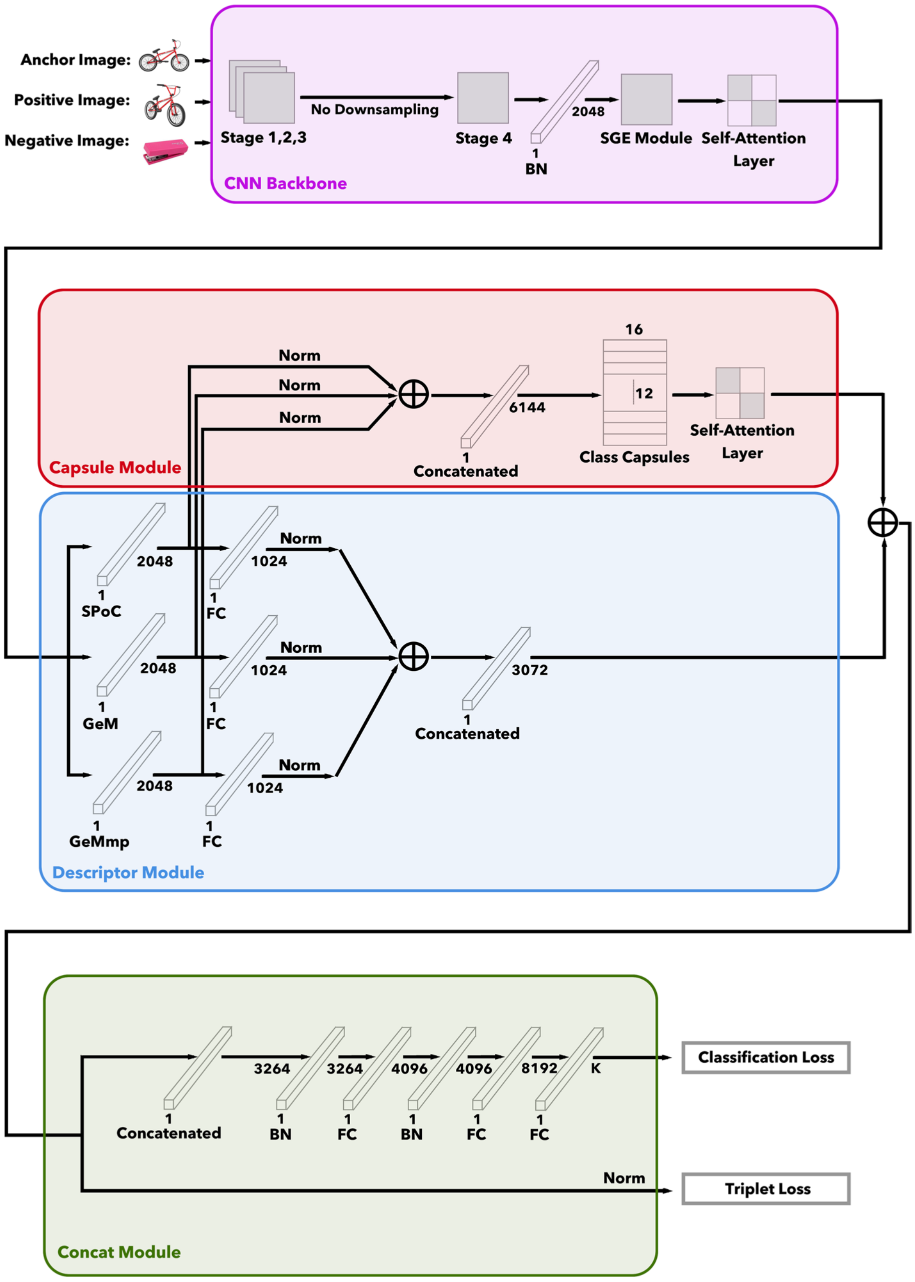

Fig. 1 The proposed model architecture. Norm, FC, BN, and K correspond to l_2-normalization, fully connected layer, batch normalization, and the number of classes, respectively. In the Descriptor module, SPoC stands for Sum Pooling of Convolutions, GeM stands for Generalized Mean pooling, and GeMmp stands for GeM with Multiple Parameters. The anchor, positive, and negative images belong to Stanford Online Products [43]

The output of the CNN backbone is sent to the branches of the Descriptor module. Each branch applies its descriptor to the given output. Following that branch's descriptor, each branch performs a fully connected layer followed by l_2 normalization. The Descriptor module is the concatenation of output feature vectors of three branches. This combined descriptor representing multiple branches is fed to the Concat module for training the proposed architecture with selected loss functions.

Generalized-Mean (GeM) Descriptor: GeM pooling layer consistently outperforms the conventional max and average pooling [34]. It provides the generalization of max and average pooling with a pooling parameter. Due to its nature of generalization, it performs better than conventional pooling methods. It reduces the dimensionality and performs normalization to obtain an image descriptor. The output of a convolutional neural network is a 3D tensor X with the shape of $C \times H \times W$, where C corresponds to the number of feature channels, H refers to the height of the feature map, and W refers to the width of the feature map for a given input image. X_c represents the set of $H \times W$ activations for feature maps $c \in \{1, \dots, C\}$. The network output contains C such activation sets or 2D feature maps. In the pooling process, the GeM pooling layer takes X as an input and generates an output vector $P^{(GeM)}(X)$. The output vector, GeM descriptor, is computed by the generalized mean of each channel in a given tensor X as follows [18, 34]:

$$P^{(GeM)}(X) = [P_1^{(GeM)}(X_1) \dots P_c^{(GeM)}(X_c) \dots P_C^{(GeM)}(X_C)]^T,$$
$$P_c^{(GeM)}(X_c) = \left(\frac{1}{|X_c|} \sum_{x \epsilon X_c} x^{o_c} \right)^{\frac{1}{o_c}}, \tag{1}$$

where o_c represents the pooling parameter. It can be a shared parameter used by all feature maps, or o_c can differ for each feature map X_c. In case of a change of pooling parameter per channel/dimension, it is called GeM with Multiple Parameters (GeMmp). The pooling parameter, o_c, can be fixed or trained. According to [5], $o_c > 1$ results in an increase in the contrast of the pooled feature map and highlights the salient features of the feature map. The output feature vector is a C-length long vector representation of an image and contains a single value for each feature map, the generalized-mean activation.
Sum Pooling of Convolutions (SPoC) Descriptor: It is a variation of the GeM pooling layer with a slight change in the exponent value, the pooling parameter [3]. It is prone to consider complete information due to its averaging nature. It also leads to better results than conventional pooling methods mostly due to the subsequent descriptor whitening. In this case, setting the pooling parameter, o_c, to 1 defines the SPoC descriptor, the generalization of average pooling, as follows [3, 18]:

$$P^{(SPoC)}(X) = [P_1^{(SPoC)}(X_1) \dots P_c^{(SPoC)}(X_c) \dots P_C^{(SPoC)}(X_C)]^T,$$
$$P_c^{(SPoC)}(X_c) = \frac{1}{|X_c|} \sum_{x \epsilon X_c} x. \tag{2}$$

3.4 Concat module

The Concat module combines the output feature vector of concatenated descriptors from the Descriptor module with the output feature vector of the Capsule module. The Concat module is composed of two branches. The output of each branch is fed into two different loss functions, as shown in Fig. 1. In the first branch, the concatenation of Descriptor and Capsule modules is followed by a series of batch normalization and fully connected layers for regularization and dimensionality reduction purposes. In the second branch, the same concatenation is fed into triplet loss simultaneously to learn similarities and differences between given image triplets after performing the l_2-normalization layer.

The motivation behind using the Concat module is to utilize the valuable information from multiple descriptor branches of the Descriptor module and the information about the objects with intrinsic spatial relationships between them and their parts from the Capsule module. Since the class capsules in the Capsule module modify the Descriptor module for classification, the concatenation of the Descriptor module and Capsule module enriches the image representations for both classification and image retrieval purposes.

Combining two modules using a single CNN backbone enables an ensemble effect and makes the proposed architecture trainable in an end-to-end manner. The concatenation of feature vectors from different modules results in different properties from each module's output, leading to better image representation.

4 Network training

During the training process, the summation of triplet loss and classification loss is utilized. Using these loss functions helps improve the model's performance by distinguishing product classes, using a classification loss, and learning similarities and differences between product classes using triplet loss.

4.1 Online triplet loss and hard negative mining

The proposed framework uses triplet loss during training, as an addition to classification loss. A triplet embedding contains three images: an anchor, a positive, and a negative image. The anchor image and positive image belong to the same class, while the anchor image and negative image belong to different classes. The triplet loss is defined over these triplets as

$$L = \max\{L_p(x_a, x_p) - L_n(x_a, x_n) + m, 0\}, \tag{3}$$

where x_a, x_p, and x_n stand for the feature embedding for anchor, positive, and negative images, respectively. L_p corresponds to the Euclidean distance between x_a and x_p while L_n corresponds to the Euclidean distance between x_a and x_n. The margin m is a hyperparameter for the triplet loss.

We minimize the triplet loss defined in (3) so that similar labeled images can be closer to each other by a margin than dissimilar ones. The triplet loss pushes $L_p(x_a, x_p)$ to 0 and $L_n(x_a, x_n)$ to be greater than $L_p(x_a, x_p) + m$. Thus, the distance between an anchor image and a positive image of the same class is minimized. In contrast, the distance between an anchor and a negative image of different classes is maximized, as shown in Fig. 2.

To train the proposed network, we need negative examples of violating this triplet condition where negative instances are closer to the anchor than the positive. Hard-negative mining is used for triplet loss for this purpose. The hard triplets contain negative images that are closer to the anchor image than the positive image, i.e., $L_p(x_a, x_p) < L_n(x_a, x_n)$. Hard negatives are used, while all positives are considered

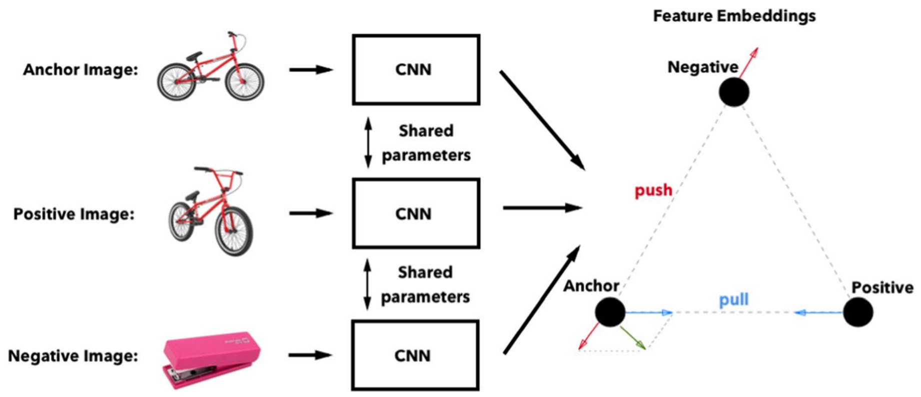

Fig. 2 The schema demonstrating how the triplet loss works for input anchor, positive and negative images. The anchor, positive and negative images belong to Stanford Online Products [43]

during the training. Online triplet mining is chosen to obtain hard-negative triplets. Online triplet mining aims to obtain triplets for each batch of inputs. Online triplet mining is much more efficient than offline triplet loss, which produces triplets at the beginning of each epoch. Online triplet mining requires computing all feature embeddings on the training set and selecting hard or semi-hard triplets.

4.2 Cost-sensitive regularized cross-entropy loss

For multi-class classification, it is possible to keep track of different kinds of errors during the neural network training process. Galdran et al. [9] introduce a cost-sensitive regularized loss. The penalization of errors avoids overfitting the neural network by keeping it more generalized. A cost matrix is used to keep track of the penalization of these errors by encoding them into it.

Based on the experiments, using cost-sensitive loss results in the model being stuck in local minima, so a cost-sensitive term must be combined with a base loss as a regularizer. As a base loss for classification purposes, the cross-entropy loss is used. A cost matrix is introduced to modify the cross-entropy loss for cost-sensitive regularization. If y_i is equal to $\hat{y}_i$, there is null cost. However, the cost increases along with the distance $\|y - \hat{y}\|$. The cost-sensitive regularized cross-entropy loss is defined as follows [9]:

$$L^{(\mathrm{CS})}(\hat{y}, y) = L^{(\mathrm{CE})}(\hat{y}, y) + \lambda \langle M^{(2)}(y, \cdot), \hat{y} \rangle , M^{(2)}_{ij} = \|i - j\|_2^2, \tag{4}$$

where λ refers to the hyper-parameter for the cost-sensitive penalty. The mechanism of label-dependent penalty is provided by encoding these costs to each row of the cost matrix and computing the L^2-based scalar product of $\hat{y}$ with the row of cost matrix M corresponding to y, which is shown as $\langle M^{(2)}(y, \cdot), \hat{y} \rangle$ in (4) [9]. The additional term in the cross-entropy loss leads to more significant penalties on the predicted label when they are farther away from a particular image's ground truth label.

5 Implementation details

We train our proposed network using batches. Each batch contains eight random product classes and four image instances from each product class to have balanced batches. If there is a product class with fewer samples, all existing samples are taken. Thus, we train our proposed network with a maximum batch size of 32. We use a pre-trained CNN backbone to accelerate the training process and initialize the weights of the other parts of the neural network using Kaiming initialization [12] for the training process. The output of the pre-trained CNN backbone results in a 14×14 sized feature map because the downsampling is discarded between Stages 3 and 4 to preserve more information in the last feature map. The pooling parameter for both GeM and GeMmp is 3. The pooling parameter was empirically determined because [5] indicates that the pooling parameter higher than 1 increases the contrast of the pooled feature map and emphasizes salient features of the feature map. Since the pooling parameter per channel is the same for GeM, the multiple parameter value is 1 for GeM and channel number (2048) for GeMmp. Each of the global descriptors corresponds to 2048-D compact image representations. We compute Recall@K for the validation dataset for early stopping and utilize the batch hard-negative triplet loss [8] with a margin of 0.3. We utilize early stopping in our experiments. The reason is that the proposed framework includes many layers, which might lead to overfitting. The aim is to extract more features to obtain richer information with these layers and avoid overfitting the training data, so early stopping is used. We use Adam optimizer [23], and the learning rate is adapted using a multi-step decay scheduler. Table 1 shows the settings used in the training process. We perform our experiments on a machine with 64 GB RAM and NVIDIA Tesla P100-PCIe GPU with 16 GB memory.

Since the number of samples for each class is limited in the current dataset, and image augmentation improves the training dataset in terms of diversity and quantity, we employ image transformation techniques as a preprocessing step. Due to memory restrictions, we resize all the images to 300×447 because most images are rectangular; the width is approximately 1.5 times of height. Then, we randomly rotate, randomly horizontally flip, randomly crop, and normalize each image for the training dataset so that the training process does not memorize. For testing, each image is center-cropped. After cropping for testing and training, the input image resolution becomes 284×423.

Table 1 Settings used in the training of the proposed architecture

Parameter	Value
Number of epochs	75
Number of epochs for early stopping	7
Learning rate for Adam optimizer	1e-4
Weight decay for Adam optimizer	5e-4
Number of capsules for the Capsule module	12
Number of routes for the Capsule module	$24 \times 4 \times 4$
Number of input channels for the Capsule module	16
Number of output channels for the Capsule module	16
Number of groups for the SGE module	64

The values of the parameters are empirically determined

5.1 Feature extraction

In the evaluation stage of the proposed framework, we utilize only a specific part of the architecture for feature extraction. Since the computational load of the Capsule module is high and the objective is to obtain descriptive and distinctive features of the images, we preferred to use the Descriptor module for feature extraction and the Capsule module for network training.

The feature extraction uses a SE-ResNet50 backbone, and the down-sampling between Stages 3 and 4 is discarded, followed by batch normalization, the SGE module, and a self-attention layer. Unlike the training process, only the Descriptor module is used for the feature extraction process, which generates the feature vector, as shown in Fig. 3. The aim is to obtain descriptive image representations for the whole dataset and given query image.

5.2 Evaluation metric

During the experiments, for evaluating the proposed framework, the Recall@K metric is chosen as the evaluation metric since it is commonly preferred for the image retrieval process. It makes it possible to compare the proposed framework's performance with existing state-of-the-art methods for image retrieval. For testing purposes, we obtain the feature embeddings from the Descriptor module as shown in Fig. 3. Using only the Descriptor module, the routing-by-agreement algorithm in the Capsule module leads to high computational cost, and the Descriptor module is the back-end of all modules.

We compute the feature embedding for each test image in the test set to calculate Recall@K. Then, we compute top-K similar images from the test set for each test image.

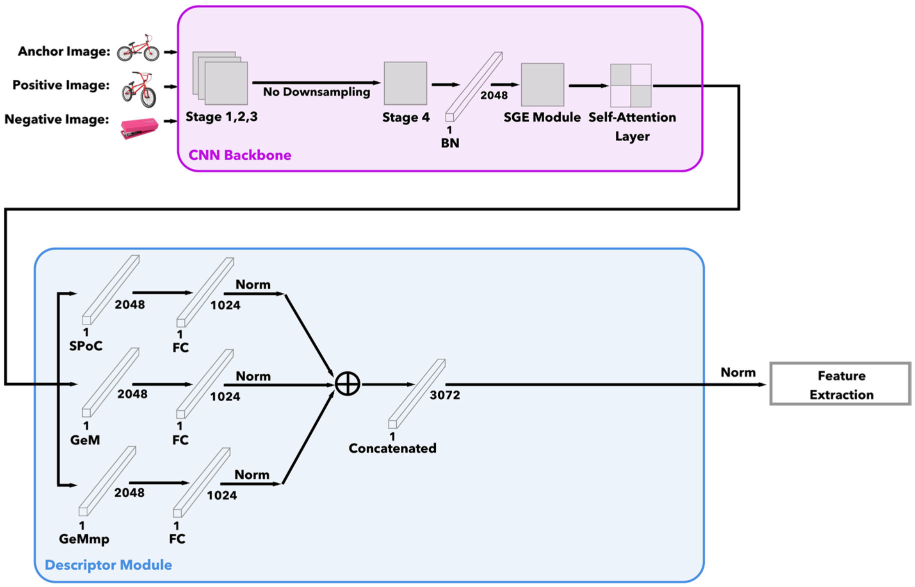

Fig. 3 The feature extraction process. Norm, FC, BN, and K correspond to l_2-normalization, fully connected layer, batch normalization, and the number of classes, respectively. The anchor, positive and negative images belong to Stanford Online Products [43]

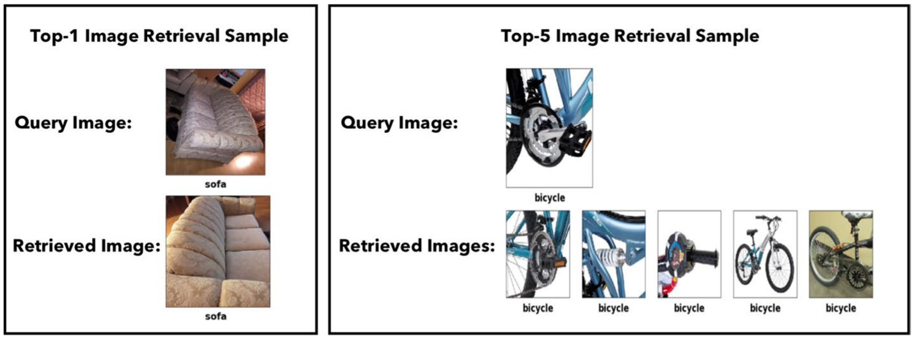

Fig. 4 Successful image retrieval examples using Stanford Online Products [43]

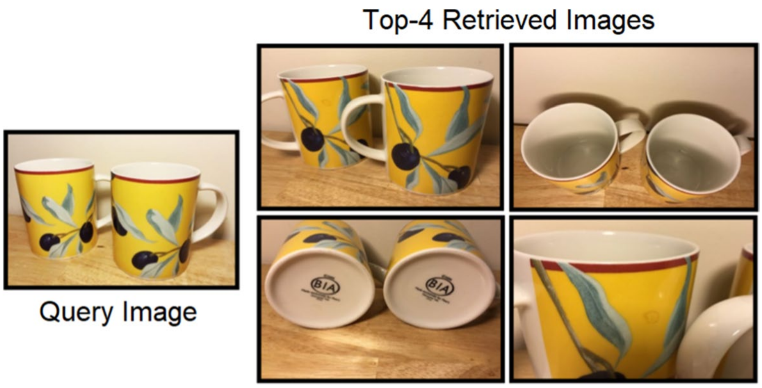

Fig. 5 Another successful image retrieval example

For each test image, the total number of predictions is incremented by one. Suppose the query image label is the same as the label of at least one image out of K retrieved images. In that case, it is a correct prediction, and the number of accurate predictions is also incremented by one. In other words, Recall@K is the number of accurate predictions among top-K divided by the total number of predictions. To compare the proposed framework with state-of-the-art techniques, we compute Recall@K, where K is 1, 10, 100, and 1000. Some successful image retrieval examples are shown in Figs. 4 and 5.

5.3 Dataset

The proposed framework is evaluated on the image retrieval dataset of Stanford Online Products [43]. We use the same training and test split with [1, 35] for a fair comparison. Considering the advantage of a large number and variety of classes in this dataset, it is one of the largest publicly available datasets [43]. Therefore, we choose Stanford Online Products to evaluate the proposed framework for image retrieval. The dataset comprises 12 superclasses: bicycle, cabinet, chair, coffee maker, fan, kettle, lamp, mug, sofa, stapler, table, and toaster. Each superclass is divided into product classes. There are 120,053 images in total and 22,634 product classes of product photos from online e-commerce websites. The 11,318 product classes with 59,551 images are utilized for training, while the remaining 11,316 product classes with 60,502 images are used for evaluation. For validation purposes, since no sample data is provided for an unbiased assessment, 7% of the training data is held back for tuning the proposed architecture's hyperparameters. During the image retrieval process, the purpose is to have the same product class for both given query images and retrieved images rather than having the same superclass.

5.4 Computational cost and memory efficiency

Rather than using an ensemble method, training and evaluation of our proposed framework in an end-to-end manner are beneficial regarding time and memory. The reason is that in an ensemble method, the number of learners for ensembling should equal the number of GPUs for individual training and evaluation. Due to the end-to-end manner of the proposed network and the usage of a shared CNN backbone, our method requires one GPU, which is also advantageous due to limited memory usage.

Regarding the computational cost and memory efficiency of the modules of our proposed model, the CNN Backbone contains a pre-trained SE-ResNet50 since a complex dataset requires more convolution layers to learn better feature embeddings. However, it results in a high computational cost and memory usage. Due to its structure, the Capsule Module has a high computational cost and memory requirement. Using capsules aims to capture the subfeatures into capsules and benefit from essential details describing images without losing the crucial spatial relationship between objects and their parts. However, since there are 10,517 product classes in the training dataset (after the validation dataset is removed from the training dataset), it is difficult to capture all features in each product class. Even with the selected number of capsules, it leads to high computational costs. The Descriptor Module has a reasonable computational cost and memory usage compared to the other modules, due to the three branches applying a descriptor to the given output, followed by a fully connected layer and an l2 normalization. The model training takes about ten hours for the Stanford Online Products Dataset, while the inference takes approximately four hours.

In terms of architectural types mentioned in the Experiments section, Architecture B leads to a higher computational cost and memory usage than Architecture A due to the extra module named Concat Module, which concatenates Descriptor and Capsule Modules. However, it has the advantage of providing richer information than Architecture A. Due to the usage of primary capsules in Architecture C, the computational cost and memory usage is the highest among all architectures. We aim to classify with respect to product classes, a significant number, i.e., 10,517. The primary capsules try to capture all the features in each product class. However, it is difficult to detect these features for that many classes with a limited number of capsules due to the high computational cost. Thus, Architecture D, which concatenates Capsule and Descriptor Modules with a Concat Module, similar to Architecture B and removes the primary capsules used in Architecture C, is acceptable compared to the other mentioned architectures in terms of computational cost, memory usage, and Recall@K.

6 Experimental results

6.1 Effect of combination of global descriptors

The proposed framework contains a Descriptor module that combines global descriptors to have descriptive image representations for input images. We conduct experiments with different configurations for the Descriptor module to assess the performance of the dataset. Based on the motivation in [18], since the best results are obtained with SPoC, GeM, and GeMmp descriptors, their combinations are experimented with in the proposed framework. The individual descriptors of SPoC, GeM, and GeMmp with 1024 embedding dimensions are combined in different configurations. Table 2 shows the performance of various combinations of selected global descriptors on Stanford Online Products. It demonstrates that the combination of SPoC, GeM, and GeMmp outperforms dual configurations with a significant performance boost. Since each global descriptor highlights different features, combining global descriptors enables a better representative embedding.

6.2 Effect of combination of modules

The proposed framework comprises two modules before concatenating them in the Concat module: the *Capsule* and *Descriptor* modules.

Although capsule networks perform superior on the MNIST dataset, they do not perform well on complex datasets such as CIFAR- 10. The reason is that the higher dimensionality of CIFAR-10 data needs more complex encodings of the images, where capsules become insufficient. Therefore, the Capsule module is evaluated together with the Descriptor module and experimented with using different combinations.

We do experiments on these modules by combining them in two different ways. In architecture type A (cf. Fig. 6), we train the Capsule module with classification loss and the Descriptor module with triplet loss. In the standard capsule networks, the output of class capsules is used to classify the given images. In architecture type B (cf. Fig. 7), we concatenate the Capsule module's output with the output of the Descriptor module using the Concat module. Then, we train the Concat module output with classification and triplet losses, similar to the proposed architecture in Fig. 1. Architecture type B works better than architecture type A because architecture type B maintains each module's properties for classification and triplet losses, leading to richer information (cf. Table 3).

This experiment indicates the importance of the concatenation of modules. Since it is important to preserve the properties of each module, the Concat module enables us to

Table 2 Recall@K for four different global descriptor combination configurations on Stanford Online Products for evaluating the effect of combining different global descriptors

Configuration	Recall@1	Recall@10	Recall@100	Recall@1000
SPoC, GeMmp	84.2	93.9	97.5	99.2
GeM, GeMmp	84.5	94.0	97.6	99.3
SPoC, GeM	84.8	94.3	97.8	99.3
SPoC, GeM, GeMmp	85.0	94.4	97.8	99.3

Each global descriptor is an output feature vector of each branch in the Descriptor module before concatenating it with the Capsule module in the Concat module

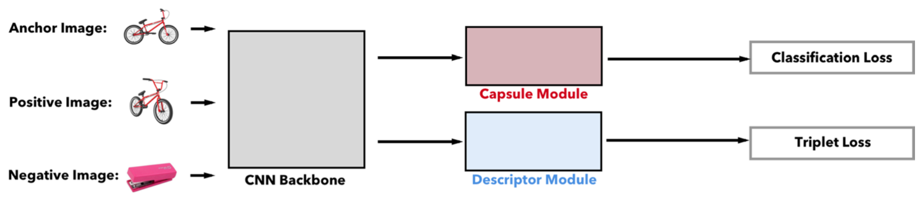

Fig. 6 Architecture type A, which does not concatenate Capsule and Descriptor modules. The Capsule module output is fed into classification loss, and the Descriptor module output is fed into triplet loss. The anchor, positive and negative images belong to Stanford Online Products [43]

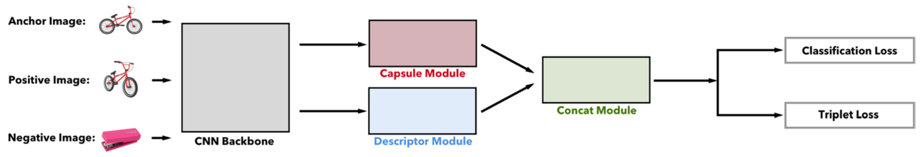

Fig. 7 Architecture type B, which concatenates the Capsule and Descriptor modules with the Concat module, and the output of the Concat module is fed into both classification and triplet loss. The anchor, positive and negative images belong to Stanford Online Products [43]

Table 3 Recall@K for two architecture types A and B on Stanford Online Products for evaluating the effect of the combination of modules

Model	Recall@1	Recall@10	Recall@100	Recall@1000
Architecture type A	81.9	92.3	96.8	99.0
Architecture type B	85.0	94.4	97.8	99.3

Figs. 6 and 7 depict Architecture types A and B, respectively

maintain them through the concatenation process. Rather than training each module to train different losses, the concatenation of the modules is utilized to maintain the diversity obtained from the two modules. Therefore, the result of this experimentation shows the effectiveness of using the Concat module.

6.3 Effect of replacing primary capsules with global descriptors

The building blocks of the proposed network are the Capsule and Descriptor modules. In the original capsule networks, the primary capsule output is fed into the class capsules to label the images. We aim to classify according to many product classes, i.e., 10,517. The primary capsules try to capture all the features in each product class without losing the spatial relationship between object and object parts. It is difficult to detect these features for that many classes with a limited number of capsules due to the high computational cost. Therefore, in architecture type D, instead of using a primary capsule layer with a limited number of capsules, the multiple branches of the Descriptor module are concatenated and reshaped to be fed into the class capsule layer to obtain a better image representation similar to the proposed architecture in Fig. 1. Figure 9 depicts the architecture type D. In architecture type C, as shown in Fig. 8, the CNN backbone output is given to primary

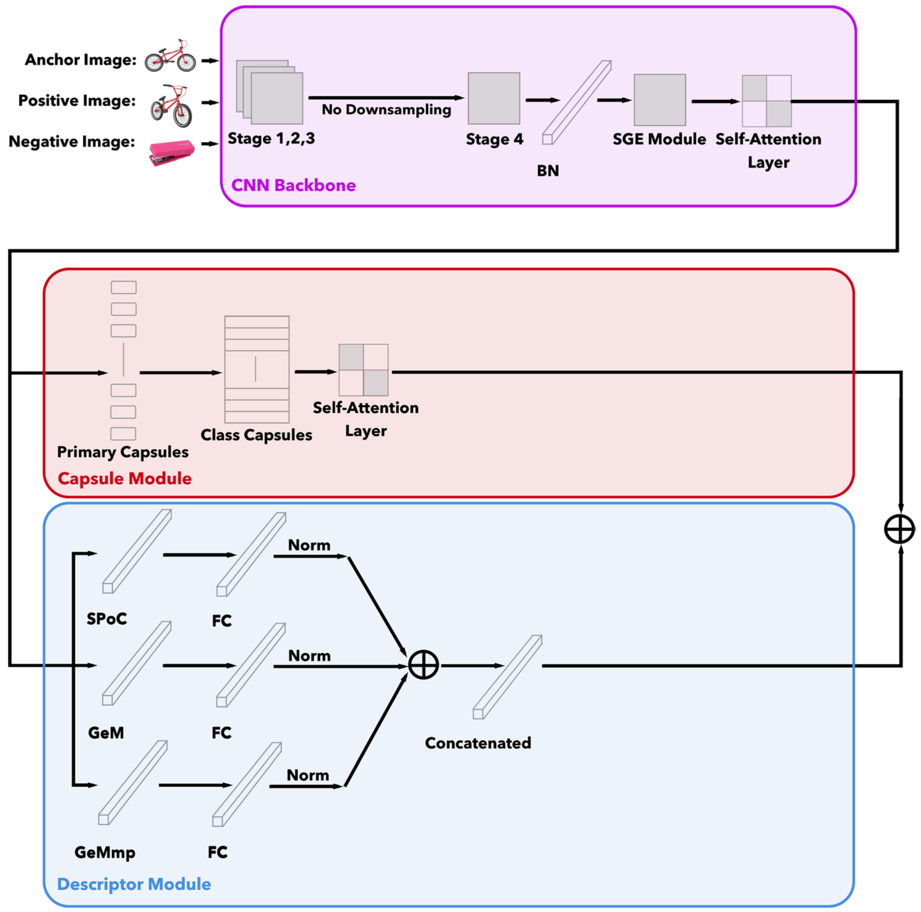

Fig. 8 Architecture type C, the CNN backbone output is given to the primary capsules, followed by class capsules. Thus, the Capsule module does not include the Descriptor module. The Concat module is not shown since it is similar in both architectures C and D. The anchor, positive and negative images belong to Stanford Online Products [43]

capsules with a limited number of capsules, so the Descriptor module is not involved in the Capsule module. We concatenate two modules at the Concat module. Architecture type D works better than architecture type C (cf. Table 4). It is because architecture type D obtains richer information by combining various descriptors. However, architecture type C cannot preserve the spatial relationships between that many objects and their parts while capturing necessary information about product images with a limited number of capsules.

6.4 Comparison with state-of-the-art methods

We compare the proposed framework's performance with state-of-the-art techniques on Stanford Online Products. Table 5 compares the Recall@K metric with previous methods

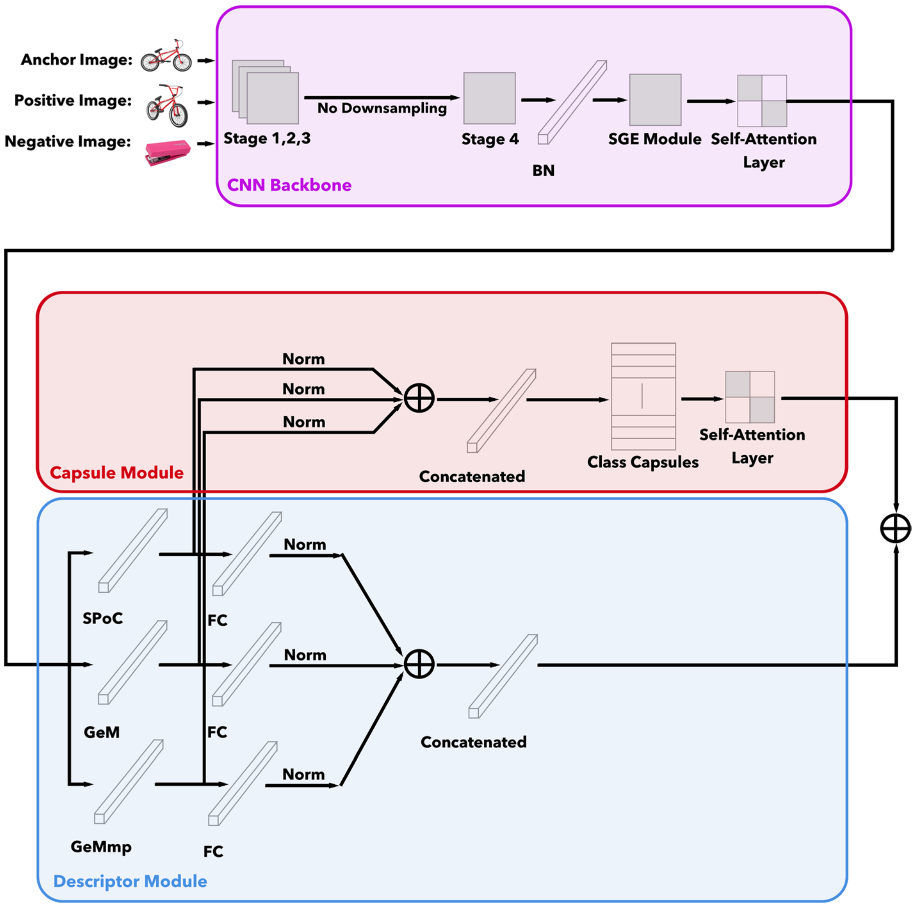

Fig. 9 The schema of architecture type D. Instead of using primary capsules, multiple branches of the Descriptor module are concatenated and fed into the class capsules. Thus, the output of the CNN backbone is not directly used as an input to the Capsule module. Again, the Concat module is not shown since it is similar in both architectures C and D. The anchor, positive and negative images belong to Stanford Online Products [43]

Table 4 Recall@K for two architecture types C and D on Stanford Online Products for evaluating the effect of replacing descriptors with primary capsules

Model	Recall@1	Recall@10	Recall@100	Recall@1000
Architecture type C	82.1	92.7	97.0	99.0
Architecture type D	85.0	94.4	97.8	99.3

Figs. 8 and 9 depict Architecture types C and D, respectively

Table 5 Performance comparison using Recall@K with previous state-of-the-art approaches on Stanford Online Products [43]

Model	Recall@1	Recall@10	Recall@100	Recall@1000
LiftedStruct [43]	62.1	79.8	91.3	97.4
N-Pairs [42]	67.7	83.8	93.0	97.8
HDC [49]	69.5	84.4	92.8	97.7
Margin [48]	72.7	86.2	93.8	98.0
A-BIER [32]	74.2	86.9	94.0	97.8
HTL [10]	74.8	88.3	94.8	98.4
ABE-8 [22]	76.3	88.4	94.8	98.2
BDB [8]	83.0	93.3	97.3	99.2
CGD [18]	84.2	93.9	97.4	99.2
Proposed Framework	85.0	94.4	97.8	99.3

such as LiftedStruct [43], N-Pairs [42], HDC [49], Margin [48], A-BIER [32], HTL [10], ABE-8 [22], BDB [8], and CGD [18] for Stanford Online Products.

Table 5 shows that our proposed framework (cf. Fig. 1) generates promising results for Stanford Online Products. As CNN backbone, HTL [10] uses BN-Inception [32] while Margin [48], BDB [8], and CGD [18] use ResNet50 [13]. A-BIER [32], ABE-8 [22], HDC [49], N-Pairs [42], and LiftedStruct [43] use GoogleNet [46]. BDB [8] utilizes an input size of 256 for testing, while the other frameworks use an input size of 224.

7 Conclusion

We proposed a framework to enhance the performance of image retrieval tasks. We build three modules on a CNN backbone, i.e., pre-trained SE-ResNet50: *Descriptor*, *Capsule*, and *Concat* modules. The Descriptor module contains three branches, each utilizing a different global descriptor. Since each global descriptor highlights the images' specific properties, it is possible to obtain richer information from each image. The Capsule module contains revised capsule networks that replace primary capsules with the concatenation of global descriptors from the descriptor. It provides essential details describing images without losing the crucial spatial relationship between objects and their parts. Thus, each image's valuable information that relates the object to its parts is preserved. Combining these modules using the Concat module during the training process enables an ensemble effect in an end-to-end manner. The summation of triplet and cost-sensitive regularized cross-entropy losses makes the proposed framework perform better since the loss considers both the similarities and differences between anchor, positive and negative images, and classification. The feature embeddings are empowered using the grouping strategy, the SGE module, to enhance sub-features parallelly and self-attention layers to explore global dependencies within internal representations of images. The Recall@K results for Stanford Online Products demonstrate superior performance compared to the state-of-the-art models with Recall@K of 85.0, 94.4, 97.8, and 99.3, where K is 1, 10, 100, and 1000, respectively.

Considering the limitations caused by the proposed model's demanding computational costs, the proposed framework's performance can be enhanced by considering the following improvements. First, various combinations of global descriptors can be experimented with for better results. Even though a single global descriptor may be inefficient for image retrieval, combining global descriptors would result in efficient image representations. We prefer a specific combination of global descriptors (SPoC, GeM, and GeMmp) in the Descriptor Module. Various combinations of global descriptors can be experimented with for better results. Second, since the capsules are insufficient to describe complex encodings of the images with limited capsules, several architectural changes can be considered on primary and class capsules to describe the image representations and classes better. Thus, it might be possible to use capsule networks directly for complex and large datasets with many classes. Finally, the proposed architecture can experiment with more complex datasets to check how expressive the model is on image retrieval tasks. Based on the experiments, the features of the proposed architecture can be arranged.

Data Availability Data will be provided to each reader on demand. Readers can request via email.

Declarations

Conflicts of interest There is no conflict of interest.

References

1. Azizpour H, Razavian AS, Sullivan J, Maki A, Carlsson S (2015) From generic to specific deep representations for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. CVPRW '15, pp. 36–45
2. Babenko A, Slesarev A, Chigorin A, Lempitsky V (2014) Neural codes for image retrieval. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer Vision - ECCV 2014. Springer, Cham, pp 584–599
3. Babenko A, Lempitsky V (2015) Aggregating local deep features for image retrieval. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV '15, pp. 1269–1277
4. Bell S, Bala K (2015) Learning visual similarity for product design with convolutional neural networks. ACM Trans Graph 34(4):10. Article no. 98,
5. Berman M, Jégou H, Vedaldi A, Kokkinos I, Douze M (2019) Multi- Grain: a unified image embedding for classes and instances. CoRR abs/1902.05509
6. Bucher M, Herbin S, Jurie F (2016) Hard negative mining for metric learning based zero-shot classification. CoRR abs/1608.07441. 1608.07441
7. Chatfield K, Simonyan K, Vedaldi A, Zisserman A (2014) Return of the devil in the details: Delving deep into convolutional nets. In: Proceedings of the British Machine Vision Conference. BMVC '14. BMVA Press, Durham, UK
8. Dai Z, Chen M, Gu X, Zhu S, Tan P (2019) Batch dropblock network for person re-identification and beyond. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. ICCV '19. Seoul, South Korea, pp. 3690–3700
9. Galdran A, Dolz J, Chakor H, Lombaert H, Ayed IB (2020) Cost-sensitive regularization for diabetic retinopathy grading from eye fundus images. In: A. L. Martel et al. (ed.) Proceedings of the 23rd International Conference on Medical Image Computing and Computer Assisted Intervention, MICCAI '20, Part IV. Lecture Notes in Computer Science, vol. 12265. Springer, Lima, Peru, pp. 665–674
10. Ge W, Huang W, Dong D, Scott MR (2018) Deep metric learning with hierarchical triplet loss. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer Vision – ECCV 2018. Springer, Cham, pp 272–288
11. Gu Y, Li C, Xie J (2018) Attention-aware generalized mean pooling for image retrieval. CoRR abs/1811.00202
12. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV '15, pp. 1026–1034
13. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR '16, pp. 770–778

14. Hinton GE, Sabour S, Frosst N (2018) Matrix capsules with EM routing. In: Proceedings of the International Conference on Learning Representations. ICLR '18
15. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR '18, pp. 7132–7141
16. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 37. PMLR, Lille, France, pp. 448–456
17. Jégou H, Douze M, Schmid C, Pérez P (2010) Aggregating local descriptors into a compact image representation. In: Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition. CVPR '10. IEEE Computer Society, San Francisco, USA, pp. 3304–3311
18. Jun H, Ko B, Kim Y, Kim I, Kim J (2020) Combination of multiple global descriptors for image retrieval. arXiv:1903.10663
19. Kalantidis Y, Mellina C, Osindero S (2016) Cross-dimensional weighting for aggregated deep convolutional features. In: Hua G, Jégou H (eds) Computer Vision – ECCV 2016 Workshops. Springer, Cham, pp 685–701
20. Kapoor R, Sharma D, Gulati T (2021) State of the art content based image retrieval techniques using deep learning: a survey 80(29561–29583)
21. Kayhan N, Fekri-Ershad S (2021) Content based image retrieval based on weighted fusion of texture and color features derived from modified local binary patterns and local neighborhood difference patterns. Multimed Tools Appl 80(21–23):32763–32790
22. Kim W, Goyal B, Chawla K, Lee J, Kwon K (2018) Attention-based ensemble for deep metric learning. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer Vision – ECCV 2018. Springer, Cham, pp 760–777
23. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) Proceedings of the 3rd International Conference on Learning Representations. ICLR '15, San Diego, CA, USA
24. Kinli F, Ozcan B, Kirac F (2019) Fashion image retrieval with capsule networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. ICCVW '19
25. Kosiorek A, Sabour S, Teh YW, Hinton GE (2019) Stacked capsule autoencoders. In: Wallach H, Larochelle H, Beygelzimer A, d' Alché- Buc F, Fox E, Garnett R (eds.) Advances in Neural Information Processing Systems, vol. 33. Curran Associates, Inc., Red Hook, NY, USA, pp. 15512–15522
26. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
27. LeCun Y, Cortes C, Burges C (2010) MNIST Handwritten Digit Database. ATT Labs [Online] 2 . Available at http://yann.lecun.com/exdb/ mnist
28. Li X, Hu X, Yang J (2019) Spatial group-wise enhance: Improving semantic feature learning in convolutional networks. CoRR abs/1905.09646. arXiv:1905.09646
29. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vision 60(2):91–110
30. Ma N, Zhang X, Zheng H-T, Sun J (2018) Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 116–131
31. Melekhov I, Kannala J, Rahtu E (2016) Siamese network features for image matching. In: Proceedings of the 23rd International Conference on Pattern Recognition. ICPR '16, pp. 378–383
32. Opitz M, Waltner G, Possegger H, Bischof H (2017) BIER - Boosting Independent Embeddings Robustly. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV '17, pp. 5199–5208
33. Özcan B, Kinli F, Kiraç F (2020) Quaternion capsule networks. CoRR abs/2007.04389. 2007.04389
34. Radenovic F, Tolias G, Chum O (2019) Fine-tuning CNN image retrieval with no human annotation. IEEE Trans Pattern Anal Mach Intell 41(7):1655–1668
35. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR '16, pp. 779–788
36. Ribeiro F, Leontidis G, Kollias S (2020) Capsule routing via variational bayes. Proceedings of the AAAI Conference on Artificial Intelligence 34:3749–3756
37. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vision 115(3):211–252

38. Sabour S, Frosst N, Hinton GE (2017) Dynamic routing between capsules. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds.) Advances in Neural Information Processing Systems. NIPS ’17, vol. 30. Curran Associates, Inc., Red Hook, NY, USA, pp. 3856–3866
39. Schroff F, Kalenichenko D, Philbin J (2015) FaceNet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR ’15, pp. 815–823
40. Simonyan K, Zisserman A (2015) Very deep convolutional networks for largescale image recognition. In: Bengio Y, LeCun Y (eds.) Proceedings of the 3rd International Conference on Learning Representations. ICLR ’15, San Diego, CA, USA
41. Sivic J, Zisserman A (2003) Video Google: A text retrieval approach to object matching in videos. In: Proceedings of the Ninth IEEE International Conference on Computer Vision. ICCV ’03, vol. 2. IEEE Computer Society, Nice, France, p. 1470
42. Sohn K (2016) Improved deep metric learning with multi-class n-pair loss objective. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R (eds) Advances in Neural Information Processing Systems, vol 29. Curran Associates Inc. Red Hook, NY, USA, pp 1857–1865
43. Song HO, Xiang Y, Jegelka S, Savarese S (2016) Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR ’16, pp. 4004–4012
44. Sun Y, Cheng C, Zhang Y, Zhang C, Zheng L, Wang Z, Wei Y (2020) Circle loss: A unified perspective of pair similarity optimization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. CVPR ’20, pp. 6397–6406
45. Sun W, Tagliasacchi A, Deng B, Sabour S, Yazdani S, Hinton GE, Yi KM (2021) Canonical capsules: Unsupervised capsules in canonical pose. In: Advances in Neural Information Processing Systems. NeurIPS ’21, vol.34
46. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR ’15, pp. 1–9
47. Tolias G, Sicre R, Jégou H (2016) Particular object retrieval with integral max-pooling of CNN activations. In: Proceedings of the International Conference on Learning Representations. ICL ’16, San Juan, Puerto Rico, pp. 1–12
48. Wu C, Manmatha R, Smola AJ, Krähenbühl P (2017) Sampling matters in deep embedding learning. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV ’17, pp. 2859–2867
49. Yuan Y, Yang K, Zhang C (2017) Hard-aware deeply cascaded embedding. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV ’17, pp. 814–823
50. Zhang H, Goodfellow IJ, Metaxas DN, Odena A (2019) Self-Attention Generative Adversarial Networks. In: Chaudhuri K, Salakhutdinov R (eds.) Proceedings of the 36th International Conference on Machine Learning. ICML ’19, vol. 97. PMLR, Long Beach, CA, USA, pp. 7354–7363
51. Zheng L, Yang Y, Tian Q (2016) SIFT Meets CNN: A decade survey of instance retrieval. IEEE Trans Pattern Anal Mach Intell 40(5):1224–1244