Cluster Computing (2025) 28:56
https://doi.org/10.1007/s10586-024-04731-w

An efficient content based image retrieval framework using separable CNNs

Sunita Rani[1] · Geeta Kasana[1] · Shalini Batra[1]

Received: 19 February 2024 / Revised: 19 August 2024 / Accepted: 23 August 2024 / Published online: 4 November 2024

Abstract
Content-Based Image Retrieval (CBIR) is a method for retrieving images based on their content rather than relying on textual descriptions or tags. Over the last decade, Deep Convolutional Neural Networks (D-CNN) based architectures have gained popularity in image retrieval. One major issue with D-CNN is that they are complex, heavy networks with high dimensional features. To overcome this bottleneck, a separable convolutional neural networks based framework has been proposed, which will reduce the complexity of the network. The proposed framework minimizes the length of the final feature vector by extracting the detailed features from the intermediate layers. This step facilitates the avoidance of abstraction of features obtained from the last layer only. The proposed technique has achieved recall and accuracy of the order of 1, indicating a high degree of relevant retrieval. For indexing and similarity matching, the approximate Nearest Neighbour Search (ANNOY) technique has been applied and optimal retrieval has been achieved when compared with the existing techniques, indicating the better performance of the proposed technique. Experimental results indicate that the proposed framework performs exceptionally well compared to the existing state-of-the-art techniques and can be easily employed in various image-related applications.

Keywords CBIR · ANNOY indexing · EfficientNetB0 · Automatic feature extraction

1 Introduction

The demand for efficient and accurate image retrieval methods has increased drastically in the era of vast digital image repositories. Content-Based Image Retrieval (CBIR) is a trans-formative approach, steering away from reliance on textual annotations or tags by harnessing the power of visual features such as colors, textures, shapes, and other intrinsic elements to determine the similarity between images. This paradigm shift enables the retrieval of images based on their content characteristics, making it a cornerstone in applications ranging from image databases to medical imaging and multimedia systems.

At the heart of CBIR lies the concept of visual similarity, where images are compared and retrieved based on their inherent visual features. This departure from conventional keyword-based searches allows for a more intuitive and nuanced exploration of image collections. Search engines equipped with image-based search functionalities exemplify the practical application of CBIR, catering to the growing need for visual information retrieval in diverse fields. As CBIR gains prominence, the challenge lies in extracting pertinent visual features in a computationally efficient and temporally optimal manner. The representation of these features as a compact feature vector is crucial, particularly in the context of large-scale image repositories. The feature vector should be concise and possess discriminative attributes to distinguish effectively among various images. This research explores and contributes to the techniques employed in extracting and utilizing visual features to enhance CBIR performance. In the pursuit of efficient and accurate CBIR, recent advancements in deep

✉ Sunita Rani
srani_phd20@thapar.edu

Geeta Kasana
gkasana@thapar.edu

Shalini Batra
sbatra@thapar.edu

[1] Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

learning have reshaped the landscape of artificial intelligence, achieving unprecedented milestones in addressing real-world challenges. Notably, deep learning techniques, with their capacity for local and automatic feature generation, have become instrumental in understanding image content. This section explores existing CNN-based approaches within CBIR, focusing on their evolution, methodologies, and applications.

Traditionally, the extraction of local features [1] in images was dominated by techniques such as Scale Invariant Feature Transform (SIFT) [2]. However, with the advent of Convolutional Neural Networks (CNNs), the paradigm shifted towards leveraging local features for instance-level retrieval. Unlike global features such as color, texture, and shape, local features capture the intricate details of objects within an image, representing a higher level of complexity that aligns with human perception. In the context of CBIR, deep learning techniques often adopt transfer learning, a paradigm where a model trained on one task is adapted to a related task. Transfer learning with pre-trained neural networks has become a cornerstone in CBIR when dealing with limited labeled data. Pre-trained models, including ResNet [3], VGG16 [4], VGG19 [4], Xception [5], originally trained on ImageNet [6] for image classification, are repurposed for image retrieval tasks. Utilizing pre-trained models as feature extractors proves effective in CBIR because they capture meaningful and hierarchical features from images. These deep CNNs learn to extract discriminative features while training on large datasets like ImageNet. The activations of neurons in the last fully connected layer serve as high-level representations or feature vectors for each image, facilitating the feature extraction process in image retrieval systems. Further enhancements involve encoding strategies such as Bag of Visual Words (BoVW) [7] and Vector of Locally Aggregated Descriptors (VLAD) [8], transforming continuous-valued feature vectors into formats suitable for traditional machine learning techniques. These encoded representations enable efficient image retrieval by leveraging standard distance metrics or serving as input features for machine learning models [9–13]. Additionally, pre-trained models can undergo fine-tuning for the target domain, adjusting their parameters to suit specific characteristics better. Techniques such as network fine-tuning allow for adaptation to the intricacies of the target dataset, enhancing the model's performance in image retrieval [12, 14, 15]. The last approach entails constructing and training a new deep network from scratch using the target dataset. While this approach offers flexibility in tailoring models to specific dataset characteristics, it demands a substantial amount of labeled data, computational resources, and expertise in neural network design. Despite its flexibility, training a deep network from scratch may be time-intensive, especially for smaller datasets, making it crucial to weigh the benefits against the resource investments.

While those mentioned above deep learning-based approaches have revolutionized CBIR, it is essential to acknowledge their inherent limitations and challenges. In the subsequent exploration of these methodologies, this work unravels the implications, challenges, and potential avenues for further innovation in CBIR. Although adept at transferring knowledge across domains, pre-trained models must be improved. These models tend to be bulky, requiring substantial storage space. Additionally, their implementation involves millions of training parameters, contributing to computational complexity. The sheer volume of parameters is summarized in Table 2, highlighting the resource-intensive nature of these models. An additional challenge stems from the high dimensions of the feature vectors obtained from the last fully connected layer. Typically of the order of 4096 or more [2], these high-dimensional vectors present practical challenges, especially when implementing systems for larger-scale datasets. The computational demands of such high-dimensional vectors can impede efficiency and scalability, impacting the feasibility of deploying content-based image retrieval systems on a broader scale. While the feature vector derived from the last fully connected layer is designed to be optimal in terms of discriminability, length, and abstraction, achieving this balance comes with inherent trade-offs. Striking the right balance between these aspects remains challenging, as optimizing for one criterion may compromise others. Therefore, a nuanced approach is required to tailor these feature vectors based on the specific requirements and characteristics of the target dataset and retrieval task.

The paper's organization will be detailed in the upcoming lines. In section 2, a comprehensive review of existing literature is provided, highlighting significant advancements and methodologies in the field. Section 3 introduces preliminary concepts essential for a thorough understanding of the research context. Section 4 details the proposed framework and experimental designs. Section 5 delineates the evaluation metrics chosen to assess the proposed framework. Section 6 delves into the experimental results and their implications. Ablation studies are discussed in Section 7. Last, section 8 offers a conclusive summary of the paper, outlining key contributions and proposing avenues for future research within the broader context of the field.

1.1 Motivation

Based on the comprehensive analysis presented in the preceding section, it is evident that DCNN-based methods can be explored further to enhance the CBIR field. However, these methods still have some improvement scope, as

discussed in Sect. 1. By strategically addressing these issues, mainly focusing on reducing complexity, retrieval time, and optimizing the length of feature vectors, the aim is to significantly contribute to the evolution of the D-CNN-based CBIR Framework.

- For efficient deployment of CBIR in resource-constrained environments, there is a need to focus on lightweight deep models with reduced computational complexity and fewer parameters. In addition, a CBIR framework designed for reduced complexity can be more easily transferable and adaptable to different domains. Models with fewer parameters can often be more generalized, effectively transferring knowledge across diverse datasets.
- Lengthy feature vectors demand higher storage requirements, increasing resource consumption. By minimizing the length of feature vectors, there is a need to optimize storage utilization, making the CBIR system more practical and cost-effective.
- Traditional CBIR systems often face challenges in terms of retrieval time, significantly as the dataset size increases. There is a need to optimize the time complexity and indexing of retrieved features to accelerate the image retrieval process, making it more efficient for real-time applications.
- In real-world applications, where efficiency and speed are paramount, a CBIR system that strikes a delicate balance between relevant retrieval and accuracy becomes indispensable. Such a system is crucial for applications across diverse domains, including healthcare, surveillance, and e-commerce.

1.2 Contribution

Motivated by the need to strike a harmonious balance between computational efficiency, storage requirements, retrieval accuracy, and time in CBIR, our research outlines clear objectives for enhancing the practical utility of the envisioned CBIR framework.

- Major focus is to reduce the overall complexity of a deep model by introducing the concept of separable convolutions. To accomplish this purpose, separable convolutional layers have been introduced to a pre-trained model, which will reduce the computational complexity in terms of a number of convolutional operations and hyper-parameters.
- To optimize the storage utilization, concise feature vectors have been obtained from intermediate layers and the last L2 normalization layer of the proposed framework, leveraging separable convolutions.
- Extracted features have been efficiently indexed to accelerate the image retrieval process. ANNOY indexing approach using Euclidean similarity has been applied to optimize retrieval time and streamline image retrieval.
- The designed framework has been tested on three datasets, which include Caltech-101 [16], Caltech-256, flower photos [17], and book covers [18]. These datasets collectively contain around 45,000 images categorized into 140 classes. The datasets are categorized based on object classes to facilitate instance-level retrieval for a focused and category-specific evaluation.
- To further demonstrate the efficiency of the suggested approach, extensive ablation study trials are performed.

Through the structured implementation of these objectives, the overall goal is to contribute to advancing CBIR systems, offering a more efficient and streamlined approach in the field of CBIR.

2 Related work

In this section, an extensive study of the existing methods and methodologies in different CBIR domains is provided. To understand the optimality of the proposed strategy, the extensive study of existing deep learning-based feature extraction techniques, the role of separable convolutions to reduce the number of convolution operations, the role of normalization techniques to enhance the discrimination power of features, the effectiveness of different indexing and similarity-based retrieval techniques has been done. CNNs have an extensive number of parameters and hyperparameters, including weights, biases, the number of layers, and processing units such as neurons, filter size, stride, activation function, and learning rate [19]. This approach aims to investigate techniques that can effectively reduce the number of hyperparameters and analyze their impact on the performance of deep models. This exploration is crucial for understanding how model complexity influences performance and identifying strategies to enhance the efficiency of deep learning architectures.

Razavian et al. in 2014 [20] conducted research on a variety of recognition tasks utilizing the OverFeat network, which was trained on ILSVRC13 for object categorization. In visual classification tasks, the network outperformed state-of-the-art systems by extracting features from the network as a generic image representation. With the exception of the sculpture dataset, the findings indicated that features derived from deep learning using convolutional nets should be the main contender for the majority of visual identification tasks, and retrieval outperformed low memory footprint techniques. They concluded that from

now on, deep learning with CNN has to be considered as the primary candidate in essentially any visual recognition task. Their work laid a foundation for the use of CNNs for a variety of tasks, including visual instance retrieval.

The study by Shiv [21] surveyed deep learning methods for CBIR from 2011 to 2020. It provided a detailed taxonomy of supervision types, networks, data types, and retrieval types. The study highlighted the evolution of deep learning models, large-scale datasets, and performance analysis. Recent models like generative adversarial networks, autoencoder networks, and reinforcement learning networks have shown superior performance. The research trend suggests that deep learning-based models are driving progress in image retrieval, with recent trends including semantic preserving class-specific feature learning, attention module, and transfer learning. Their study provided a thorough summary of the research area, including the primary developments, methodologies, and trends. This facilitates comprehension of the field's broader context and history, which is helpful to benchmark new approaches.

In order to close the semantic gap in CBIR systems, Wan et al. [22] investigated the possibilities of deep learning techniques. Their study found that the use of Convolutional Neural Networks, a cutting-edge deep learning technique, for CBIR tasks in various scenarios. The study presented the importance of bridging the semantic gap between high-level semantic concepts perceived by humans and low-level picture pixels collected by machines. It also discovered optimistic findings and offered recommendations for future research. Despite obtaining the satisfactory values of the mAP metric, the low recall values indicated the degree of relevance of the retrieved images was not satisfactory.

Heba et al. [23] used Alexnet [24], one of the very first pre-trained models with 8 layers only. They obtained features for each image from layers FC6 and FC8 of the Alexnet, and a weighted sum of these two vectors was calculated to increase the efficiency of the features extracted from FC8 by introducing partial support of the FC6 layer. This study deployed the AlexNet model for feature extraction, which, being bulkier with a high number of parameters, is unsuitable for resource-constrained environments.

Ahmed Ali [13] used pre-trained ResNet18 and SqueezNet as feature extractors. He divided the CBIR process into online and offline processes. The offline process extracts the features of the query and database images, while the online process contains the phases for similarity and retrieval. During the similarity phase, the Euclidean distance was used to compare the feature vectors of the query image and database features. Among both architectures, ResNet18 performed better than SqueezeNet. Despite the comparable performance, both these architectures have drawbacks. ResNet18 contained residual blocks that added complexity to the model in terms of increased operations. SqueezeNet was designed with minimal parameters, which reduced its limit, flexibility, and adaptability to different tasks.

Sikandar et al. used ResNet50 [3] and VGG16 [4] pre-trained architectures and one machine learning model, K-nearest neighbors. Resnet50 and VGG16 were combined as feature extractors of images to create the final feature vector. The length of the final feature vector is 2560 after the feature fusion of Resnet50(2048) and VGG16(512). K-NN clustering was performed on the extracted features and retrieved similar images close to a query image [10]. While achieving perfect precision, the number of parameters and length of the feature vector of this deployment are considerably high.

Subhadip et al. in 2020, also proposed the use of pre-trained models. They took activations from the different layers of pre-trained models and concluded that initial layers are depictions of edges only; the activations of deeper layers can represent the more detailed high-level features. They also compared the performance of different network architectures DenseNet [25], InceptionResNetV2 [26], InceptionV3 [26], MobileNetV2 [27],NasNet Large [28], ResNet50 [3], VGG19 [4], Xception [5]. In a comparative study of these models, they found InceptionResNetV2 to be best for image retrieval as it achieved a significant 96.115% mean average precision value [11]. However, the length of the feature vector is quite high.

Alappat et al. [12] utilized the MS-RMAC method for feature extraction. They employed the InceptionV3 [29] model and extracted activations from its last fully connected layer. The activations were divided into branches, and separate feature extraction was performed on each branch, resulting in multiple features flattened into vectors. These individual vectors were then combined to obtain a single, unified feature. The experiments were conducted on the CUB200-2011 dataset. Their experimentation achieved a training accuracy of 99.46% and a validation accuracy of 84.56%. They further improved the validation accuracy to 88.89% by employing three-branched global descriptors. The MS-RMAC features are entirely based on the branching of the global descriptor of InceptionV3. Due to its intricate design, optimizing IncepionV3 for specific tasks is quite challenging.

The use of pre-trained models is not limited to feature extraction. Recently, Suneel et al. have proposed a modified- VGG16 to target robust features. They have reduced the dimensions of the features vector using principal component analysis M-VGG16+PCA. They found a huge difference of 88% in the feature vector dimensions with and without PCA. Low dimensional features lead to less memory utilization, which is highly desirable in the case of

Table 1 Summary of related work

Year	Model	Parameters	Indexing/similarity	Feature vector	References no.
2019	Alexnet	–	Cosine	–	[23]
2020	InceptionResNetV2	–	Manhattan, Euclidean	–	[11]
2021	InceptionV3	–	L2 Norm	–	[12]
2021	ResNet18, SqueezeNet	–	Euclidean	–	[13]
2023	ResNet50, VGG16	23,587,712 and 14,714,688	KNN, Euclidean	2560	[10]

Table 2 Summary of existing feature extractor models

Model	Parameters	Feature vector length
Alexnet	28,823,912	1000
InceptionResNetV2	54,336,736	98,304
InceptionV3	21,802,784	131,072
ResNet18	11,708,328	1000
SqueezeNet	421,098	1000
ResNet50, VGG16	23,587,712	2048
VGG16	14,714,688	25,088
EfficientNetB0 (without top layer)	4,049,571	62,720

large datasets. Low dimensional features can also be indexed efficiently, reducing the retrieval time considerably [30]. However, in certain situations, the reduced feature space may be incapable of representing the required features of the images.

Faraj et al., in their work, focused on effectiveness rather than efficiency. They developed a hybrid CBIR technique with two layers of search, called bi-layers. These two layers contributed to obtaining both local and global features. The first layer aimed to eliminate dissimilar images by comparing all photos in the dataset to the query image using the SURF Features local feature descriptor. The second layer compared the query image to the pictures attained in the first layer based on extracting global features like shape, texture, and color. The proposed work introduced the idea of producing cases to make the system more dynamic and increase the precision rate [31].

Abdul et al. proposed a hybrid CBIR technique with two layers of filtering. The first layer used the Bag of Features (BoF) technique to compare the query image to all images in the database, resulting in images closer to the query image. The second layer compared the query image to the retrieved images using texture and color-based features. The effectiveness of the proposed CBIR approach was increased by creating four cases for implementing the system on the entire image or parts separately. The researchers used the Corel-1 k database and found that the proposed two-layer strategy outperforms existing state-of-the-art approaches in precision rate [32].

Vishwanath et al. proposed an interactive content-based image retrieval system that used variable compressed convolutional info neural networks (VCCINN) to retrieve images in response to image queries. The variable info algorithm optimized the weight of the neural network, and matching activity was done by recursive density matching. The interactive technique eliminated irrelevant images based on user feedback; only the relevant images were retrieved. The overall retrieval performance in Caltech-101 (dataset 1) and Inria holiday (dataset 2) are 98.17% and 99%, respectively. The differential learning-based introduced image retrieval approach outperforms several existing methods regarding image similarity and retrieval speed [33].

The details of related work are summarized in Tables 1 and 2. The number of parameters and length of the final feature vector are crucial for the large and scalable CBIR models.

3 Preliminaries

3.1 Separable CNNs

A separable convolutional neural network [34] employs a combination of depth-wise convolutions and point-wise convolutions to reduce the number of parameters and computational complexity while maintaining model performance. This architecture is designed to enhance efficiency in terms of both computation and memory usage. It

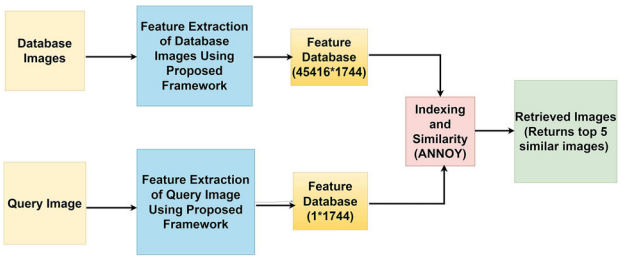


Fig. 1 CBIR process using proposed framework

derives its name from the fact that in a convolutional operation for a kernel size of (k,k), a total k^2 number of operations are required per pixel. The separability property states that a $k \times k$ kernel can be separated into two 1D horizontal and vertical convolutions, reducing the k^2 number of operations to only 2k per pixel. The separability property inherent in depth-wise separable convolutions contributes to the acceleration of the convolutional process by reducing the number of convolutional operations per pixel. This property is a key factor in enhancing the efficiency of the D-CNN architecture. Our approach would be to eliminate the softmax layer of EfficientNetB0 [35] and add a layer of separable CNN, reducing the computational complexity compared to standard convolutional layers [36]. It achieves this by decomposing the standard convolution operation into two separate steps: depthwise convolution and pointwise convolution (Fig. 1).

1. In Depthwise convolution, the input channels (or feature maps) are convolved separately with their respective filters. The number of output channels remains identical to the number of input channels. Depthwise convolution, as a component of depthwise separable convolutions, performs a spatial convolution independently for each input channel [36].
2. Pointwise convolution is a 1x1 convolution that is applied after the depthwise convolution. It projects the depthwise convolution outputs into a new space by applying a linear transformation. This step allows cross-channel interactions and enables the network to learn more complex representations [37].

Although these operations have been implemented by a few pre-trained models such as Xception [5], Mobilnets [38], and EfficientNets [35] itself to the best of our knowledge, it is being employed for the first time for image retrieval.

3.2 Base architecture

The EfficientNets [35] is a family of models with versions B0 to B7 that can achieve much better accuracy and efficiency than previous ConvNets such as Alexnet [24], ResNet [3], VGGNet [4]. EfficientNets [35] considers model scaling along the depth, width, and resolution with fewer number of training parameters [35]. Therefore, EfficientNetB0 has been used as a base model for its transfer learning capability to learn discriminative features. Along with this, the following points have been considered to make use of EfficientNetB0 for retrieval tasks:

1. Improved Accuracy-Model Size Trade-off: EfficientNetB0 achieves better balance among model accuracy and model size compared to previous models. It employs a compound scaling method that concurrently scales model depth, width, and resolution. This approach allows EfficientNetB0 to achieve higher accuracy than smaller models while being more efficient than larger models.
2. Efficient Use of Computational Resources: EfficientNetB0 is designed to optimize the utilization of computational resources. Scaling the model parameters based on a set of coefficients achieves better performance than other models of similar size. This efficiency makes EfficientNetB0 adaptable to settings with limited resources, like mobile devices or edge computing scenarios.
3. Transferability of Learned Features: EfficientNetB0 is pre-trained on large-scale datasets like ImageNet, which enables it to learn generic features that can be transferred across diverse computer vision tasks. By leveraging the learned representations from a large and diverse dataset, EfficientNetB0 is an effective initial stage for transfer learning, diminishing the necessity for extensive training on datasets specific to a particular task.
4. Scalability to Larger Models: The EfficientNet architecture is designed to scale up to larger variants (EfficientNetB1, B2, etc.) while maintaining the same efficiency principles. This scalability allows

researchers and practitioners to choose models of different sizes based on their specific needs without sacrificing performance or efficiency.

5. Versatile Applications: EfficientNetB0's success across diverse computer vision tasks highlights its role as a robust and adaptable architecture in the field. Its ability to achieve a favorable balance between model size and performance makes it a valuable asset in various applications. Its versatility makes it well-suited for a broad spectrum of applications and enables researchers and practitioners to leverage its efficiency and accuracy benefits across different tasks.

In the proposed approach, the base model has been modified by using separable convolutional neural networks on top of EfficientNetB0 to minimize the number of convolution operations. Further, using a pre-trained model as a feature extractor reduces the large amount of overhead when there is a resource constraint. It is very difficult to gather large amounts of labeled data for supervised models. Referring to an example of an ImageNet dataset, deep learning models are inherently domain-specific, and the process of curating large datasets, such as ImageNet, demands significant effort and resources. Models trained on these extensive datasets often achieve state-of-the-art performance, demonstrating considerable accuracy and surpassing benchmark results. This leads to the use of transfer learning capabilities of a model that allows the knowledge of one pre-trained model on massive datasets to be reused by other models [39]. Secondly, retrieval performance using a pre-trained network was affected by domain shifts, i.e., a model trained on the source dataset for the classification task was transferred to a new dataset for the retrieval task.

Feature extraction is the primary task in image retrieval. The features that are extracted are supposed to possess the discriminative ability. To extract meaningful embeddings, it is essential to choose a robust base model. In the proposed approach, EfficientNetB0 has been used as the base model for feature extraction due to its underlying features. The usual methods to scale a model were depth, width, or resolution scaling. EfficientNet networks were the first to consider all three aspects, which was called the compound scaling method [35]. These are the scaled-up versions of ConvNets with better accuracy and efficiency. The length of the feature vector generated by EfficientNetB0 is (7,7,1280), the smallest among EfficientNet family networks.

4 Methodology

The transfer learning approach is employed to leverage the capabilities of pre-trained convolutional neural networks (CNNs) for the feature extraction task. The model is initialized with a pre-trained architecture mentioned in Sect. 3.2 and removes its last softmax layer, rendering it suitable for extracting image features. The second layer, the top activation layer, is the starting point for subsequent modifications. The detailed methodology is discussed in Algorithm 1.

Algorithm 1 Image retrieval using proposed framework

1: **Initialize Base Model:**
- Import EfficientNetB0 pre-trained model without the top classification layer and set input shape of the model to (224, 224, 3).

2: **Modify Model Architecture:**
- Extend the base model by adding layers for enhanced feature extraction:
- Add SeparableConv2D with 128 number of channels, kernel size 3x3 and relu activation function.
- Apply GlobalAveragePooling2D to reduce spatial dimensions.
- Add a Dense layer to produce embeddings of a specified size.
- Implement an L2 normalization layer to normalize embeddings to the specified size (256) Figure 4.

3: **Compile the Model:**
- Define the optimizer Adam with a specific learning rate of 0.0001 and loss function sparse categorical cross-entropy.

4: **Import and Prepare Dataset:**
- Read and organize image filenames and assign categorical labels to each image category.

5: **Data Preprocessing:**
- Resize all images to (224, 224, 3) to fit the model input size.
- Convert images and their corresponding labels into numpy arrays.

6: **Split Data to Test and Training Sets:**
- Split the dataset into training and testing sets with a test size of 0.3 (e.g., 70% training, and 30% testing).

7: **Scale Data:**
- Normalize pixel values of training and testing images by scaling pixel values to the range [0, 1]).

8: **Model Training:**
- Fine-tune the model with a batch size of 32 for 10 epochs implementing early stopping to prevent overfitting and restore the best model weights based on validation performance.

9: **Feature Extraction from Dataset:**
- Extract features from selected layers, i.e, block2a_expand_activation, block4a_expand_activation, block5a_expand_activation and block6a_expand_activation of the trained model.
- Combine features from multiple layers into a single feature vector representation.
- Apply L2 normalization to ensure feature vectors are of unit length.

10: **Build Index for Fast Retrieval:**
- Use ANNOY indexing to create an index structure for feature vectors derived from the dataset to 200 building trees for optimal retrieval performance.

11: **Preprocess Query Images:**
- Resize query images to (224, 224, 3).
- Normalize query images using the same scaling method as the training and testing data.

12: **Extract Features from Query Images:**
- Use the trained model to extract features from query images, point (9.

13: **Retrieve Nearest Neighbors:**
- Perform a nearest neighbor search in the indexed feature space using features extracted from query images.
- Retrieve a top 5 number of nearest neighbors based on Euclidean similarity metrics.

14: **Plot the Results:**
- Plot retrieved images along with their similarity scores or distances and save values of evaluation metrics to a file.

4.1 Proposed model

After removing the softmax layer, a small augmentation network is integrated. This augmentation network comprises distinct layers designed to enhance the model's feature representation with fewer hyperparameters.

4.2 Augmentation network

By integrating the augmentation network after removing the softmax layer, the aim is to tailor the pre-trained model for our specific feature extraction objectives, enriching its representational capacity for subsequent tasks. The key components of the augmentation network include:

4.2.1 Separable convolution 2D

The introduction of an additional layer serves the purpose of augmenting the depth and non-linearity of the model. By incorporating this extra layer, the model gains the capacity to capture more intricate patterns and features within the data. The increased depth enables the neural network to learn hierarchical representations, allowing it to discern and understand complex relationships among features. As mentioned, separable convolutions are a variant of standard convolutions that split the operation into depthwise convolution and pointwise convolution as depicted in Fig. 2. The output of the pre-trained EfficientNetB0 base model is used as the input to this layer, i.e., the additional Separable Convolutional layer operates on the feature maps generated by the EfficientNetB0. The input shape (None, 2, 2, 128) suggests that the layer takes an input with a shape of (batch size, 2, 2, 128) and takes 175,488 as trainable parameters in the layer. This layer generates an output with the same dimensions as the input and establishes a connection to the top_activation layer. Through this linkage, the information flow is maintained, preserving the spatial characteristics and structure of the input.

The additional Separable Convolutional layer acts as a feature transformation layer on top of the EfficientNetB0 features, allowing the model to learn more abstract and higher-level representations specific to the task at hand. It adds more capacity to the model and enables it to adapt the learned features to the specific data and task to improve performance. Figure 4 shows the layers of modified EfficientNetB0 with respective layers.

4.2.2 Global average pooling 2D

This layer receives input from the preceding layer, separable_conv2d, and generates an output with a shape of (None, 128). Remarkably, it introduces no additional parameters. The main objective of global_average_pooling2d is to condense the spatial dimensions of the input while preserving crucial information and features. Conventionally, this operation is applied following one or more convolutional layers in a CNN. The network aims to capture essential patterns across the entire feature map by performing global average pooling, aiding in higher-level feature extraction and abstraction. These convolutional layers result in feature maps with spatial dimensions (height and width) that represent the activations of various filters at different locations in the input image. The Global Average Pooling 2D layer functions by calculating the

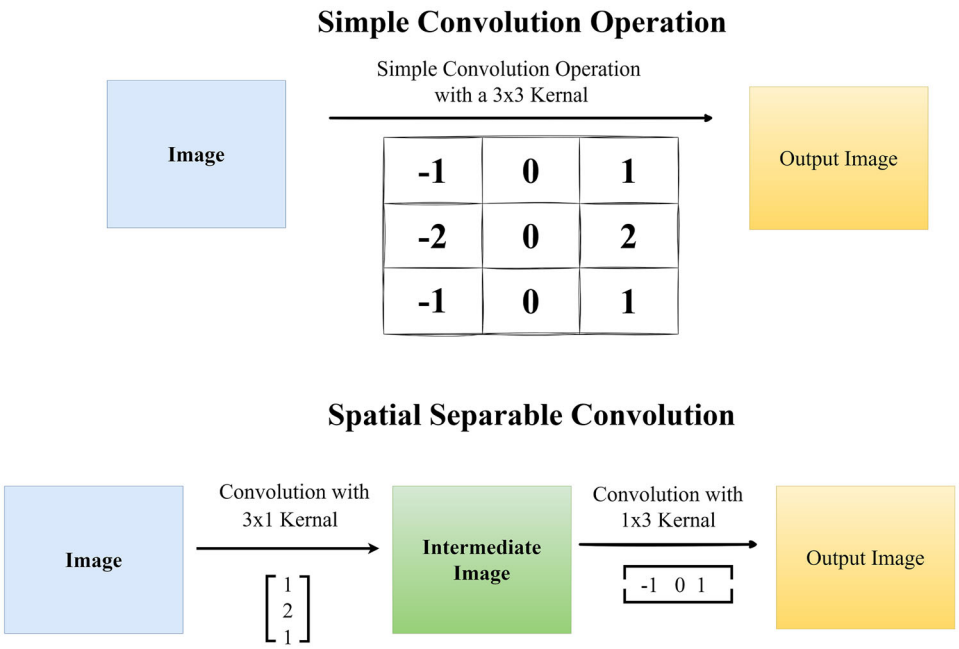


Fig. 2 An example of simple convolution operation and spatial separable convolution

average of each feature map independently across all spatial locations. This operation condenses the spatial dimensions, producing a global summary of each feature. This pooling technique effectively retains important information while significantly reducing the number of parameters, contributing to a more computationally efficient and generalized model. For each channel, it calculates the average value across the entire feature map. This contrasts traditional convolutional layers that use filters to capture spatial patterns (Fig. 3).

The output of the Global Average Pooling 2D layer is a tensor with a reduced spatial dimension. Typically, it reduces the height and width to 1x1. The number of channels (depth) in the output tensor remains the same. This 1x1 feature map retains information about the presence of various features in the original input. Global Average Pooling can provide spatial invariance to small translations in the input. As it takes the average over the entire feature map, the precise spatial location of a feature becomes less critical. This technique efficiently reduces the number of parameters in a neural network. This operation contributes to a more streamlined model with fewer parameters, which can benefit computational efficiency and model generalization. It allows a CNN to produce a fixed-length vector representation of an image's features, making the model suitable for feature extraction.

4.2.3 Dense layer

The Dense layer, a fully connected layer, is a type of neural network layer in which each neuron is connected to every neuron in the previous layer, i.e., global_average_pooling2d. This means that all the output values from the preceding layer contribute to the inputs of each neuron in the Dense layer. During training, the weights and biases of the dense layer are updated through backpropagation to minimize the loss function. The layer learns to adapt its parameters to make accurate predictions based on the training data. It has 33,024 parameters, with the output shape (None, 256). The embedding size has been restricted to 256 only to optimize the length of the feature vector. In neural network terminologies, the first dimension, denoted as 'None', represents the batch size, and it can vary based on the number of input samples during training or inference. The second dimension represents the number of neurons or units in this layer. Each of these 256 neurons produces an output. The number of parameters is calculated using the formula:

$$\begin{aligned}\text{Num\_of\_para} = (\text{Num\_of\_neurons\_previous\_layer}) \\ \times (\text{Num\_of\_neurons\_in\_dense\_1}) + \text{Num\_of\_biases}\end{aligned} \tag{1}$$

Num_of_neurons_previous_layer is 128 in this case and *Num_of_neurons_in_dense*_1 are 256, which is the number of weights of dense_1 and comes out to be 32768. A neural network layer's total number of parameters is the sum of the weights and biases. This case has 32,768 weights and 256 biases, resulting in 33,024 parameters. Dense layers play a crucial role in learning complex patterns and relationships within the data. These layers contribute to the model's ability to capture higher-level features and perform non-linear transformations on the input data. This enables the neural network to learn and represent intricate and abstract patterns in the data. The output of the dense layer can be used for different tasks. In a classification problem, this layer may be followed by a softmax activation function to produce class probabilities. In regression problems, it can be used to predict numerical values. Here, since the feature extractor model is being used, the dense layer is followed by the normalization layer.

4.2.4 L2 normalization

L2 normalization is a technique employed to normalize the output of a neural network layer, ensuring that the feature

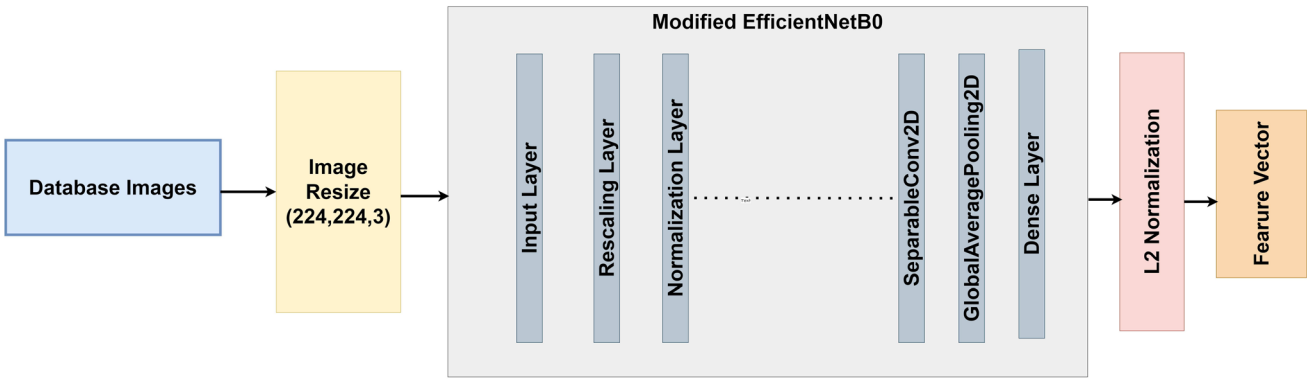


Fig. 3 Graphical representation of proposed framework along with augmentation network

vectors consistently have a uniform length. This layer takes input from a dense layer and produces an output of shape (None, 256). Through this layer, the size of the feature vector has been customized from 1280(the length of the feature vector produced by the second last layer of EfficientNetB0) to 256. L2 normalization, or Euclidean normalization, is commonly used in machine learning and deep learning to normalize vectors. It scales each element of the vector by dividing it by the L2 norm of the vector, which is the square root of the sum of the squares of all elements. In the context of neural networks, L2 normalization is often applied to the output of a layer to ensure that the feature vectors have the unit length required for image retrieval. L2 normalization is widely used in image retrieval tasks because it helps to overcome the scaling issue and ensures that the comparison is more robust across different feature vectors (Fig. 4).

The L2 normalization scales a vector x by dividing each element by its L2 norm. The L2 norm (Euclidean norm) of a vector x = [x1, x2,..., xn] is calculated as the square root of the sum of the squares of its elements. Mathematically, the L2 normalization of a vector x is given by:

$$L2\_normalized\_x = x/||x||2 \tag{2}$$

where,

$L2\_normalized\_x$ is the L2 normalized vector.

$x = [x1, x2, ..., xn]$ is the original vector.

$||x||2$ is the L2 norm of the vector x, calculated as:

$$||x||^2 = sqrt(x1^2 + x2^2 + ... + xn^2) \tag{3}$$

In formula form for an n-dimensional vector $x = [x1, x2, ..., xn]$, the L2 normalization is:

$$L2norm\_x = [x1/sqrt(x1^2 + x2^2 + ... + xn^2), x2/sqrt(x1^2 + x2^2 + ... + xn^2), ...xn/sqrt(x1^2 + x2^2 + ... + xn^2)] \tag{4}$$

The L2 normalization ensures that the vector's magnitude becomes 1 while preserving the direction of the original vector. This can prevent the model from being sensitive to the scale of the input features and improve the model's generalization ability.

L2 normalization is often used with embeddings or feature vectors that are expected to be compared using similarity metrics. Overall, this layer helps ensure that the feature vectors produced by the model are well-scaled and suitable for similarity-based tasks without introducing additional learnable parameters (Fig. 5 and Table 3).

4.3 ReLU activation

The proposed model uses non-linear *ReLU*(Rectified Linear Unit) activation at both separable and dense layers of the modified model. *ReLU* activation is a widely used activation function in neural networks and deep learning models. A simple yet effective non-linear activation function introduces non-linearity to the model. It's known for its ability to train very deep networks effectively. *ReLU* activation is a fundamental and popular choice in deep learning due to its simplicity, non-linearity, and effectiveness in training deep neural networks. The *ReLU* function is defined as follows:

$$f(x) = \max(0, x) \tag{5}$$

The function f(x) returns the input value x if positive and zero if negative.

4.4 Computation of compilation and loss

The modified model is fine-tuned against the target dataset using compilation and computation of the loss function. The Adam optimizer is a popular optimization algorithm that adapts the learning rate during training. The loss or loss function is a vital component of the training process. It quantifies how well the model performs on the training data by measuring the difference between the predicted and target outputs. The goal of the training process is to

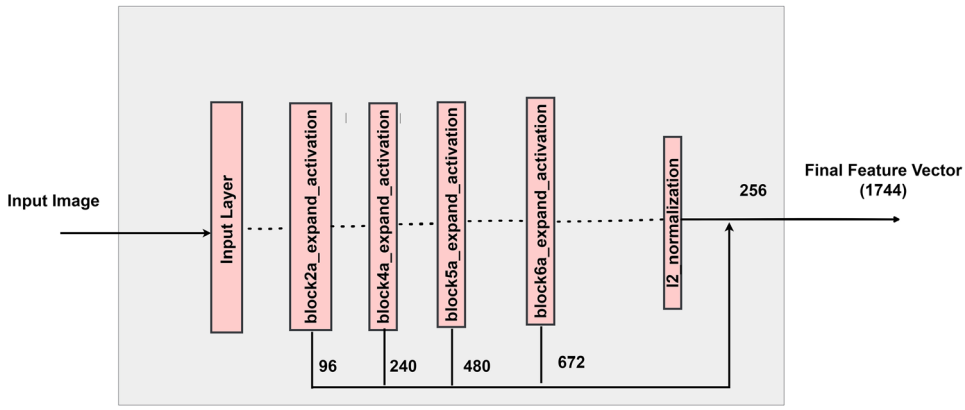


Fig. 4 Intermediate layers of proposed feature extractor model

Fig. 5 Number of Parameters of Proposed Framework

```
separable_conv2d (SeparableConv2D)   (None, 5, 5, 128)    175488   ['top_activation[0][0]']
global_average_pooling2d (GlobalAveragePooling2D)   (None, 128)    0   ['separable_conv2d[0][0]']
dense_1 (Dense)   (None, 256)    33024   ['global_average_pooling2d[0][0]']
l2_normalization (L2Normalization)   (None, 256)    0   ['dense_1[0][0]']
==========================================================================================
Total params: 4258083 (16.24 MB)
Trainable params: 4216060 (16.08 MB)
Non-trainable params: 42023 (164.16 KB)
```

Table 3 Comparison of network parameters and feature vector of proposed framework

Model	Parameters	Length of feature vector
Sikandar et.al. [10]	23,587,712 and 14,714,688	2560
Proposed framework	4,258,083	1744

minimize this loss function so the model's predictions become more accurate. The model aims to reduce this loss function during training to improve its predictions. Figure 6 depicts the proposed model's training and validation loss.

4.5 Proposed method of feature extraction using intermediate layers

The intermediate layers of different blocks of a deep network can capture meaningful and abstract features. Therefore, the different intermediate layers of Modified-EfficientNetB0 are used to construct a final feature vector. As shown in Fig. 4, these layers are block2a expand activation (96), block4a expand activation (240), block5a expand activation (480), and block6a expand activation (672). The features of all these layers, along with the last layer that is l2_normalization(256), have been concatenated, making the final feature vector 1744. The layers block2a_expand_activation and block4a_expand_activation are feature extractor layers from the initial blocks of the model, while block5a_expand_activation and block6a_expand_activation are from the deeper blocks of the model. These layers provide a balance between capturing meaningful content and avoiding excessive abstraction. The model's selection criteria include layer depth, feature diversity, and content/abstraction balance. The first layers collect lower-level elements such as edges and textures, whereas the deeper layers catch higher-level abstract properties. The goal is to generate a rich and diversified feature vector that combines fine-grained features with broader contextual information, hence increasing the model's robustness and accuracy. Balancing content and abstraction is critical, and the chosen layers provide an ideal balance of detailed and abstract information. The model accepts that various configurations, such as focusing more on shallow or deeper layers, could be investigated depending on the application or dataset characteristics. The current strategy provides a solid basis, but alternate layer selections may produce different results depending on unique requirements or aims.

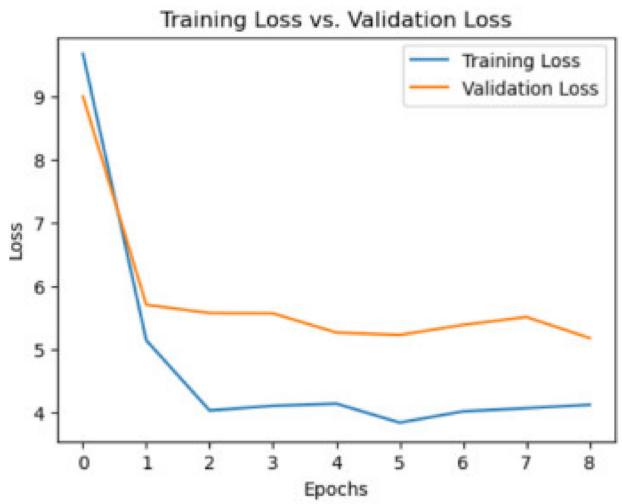


Fig. 6 Training v/s Validation loss of Proposed Framework

4.6 Feature vector constructed using proposed framework

The proposed methodology shares the idea of utilizing concatenated features but with a distinctive approach. Instead of using disparate pre-trained models such as ResNet50 and VGG16 [10], the proposed approach focuses on a Modified-EfficientNetB0 model for feature extraction. Using pure EfficientNetB0 as a feature extractor without a softmax layer is not feasible because the length of the final

feature vector is of very high dimensions and is not recommended for a large dataset. The choice of the EfficientNet architecture is motivated by its efficiency and effectiveness in image-related tasks. A key departure from Sikandar's approach [10] is that the proposed technique emphasizes obtaining concatenated features from intermediate layers. This allows us to capture diverse and multi-level representations of image content.

The selection of intermediate layers is a critical aspect, and the aim is to identify layers that offer optimal discriminative features for the CBIR task. The last layer of the feature extractor model is a BatchNormalization layer with a shape of (None, 7, 7, 1280). 7 and 7 represent the feature map's spatial dimensions (height and width), and 1280 represents the number of channels or features in the feature map. After flattening, the feature vector of length 7*7*1280 is obtained, which is 62720. Although precision is 100% for every image in any database using this approach, such a considerable feature vector length is not desirable for any retrieval task. Instead, the feature extractor mentioned in section 4 is used to construct the final feature vector.

The output shape of the feature vector obtained from the second last layer of EfficientNetB0 is (None,1280). To reduce the space complexity, the length of the feature vector has been customized to 256 using the L2 normalization layer. The last L2 normalization layer and the few intermediate layers of the proposed framework are used to obtain features 4. Therefore, the total length of the final feature vector is 1744. There are 45,416 images in our dataset. The memory requirement is calculated (Eq. 6) using the formula:

$$\begin{aligned}\text{Memory Requirement(B)} &= \text{Number of Images} \\ &\times \text{Feature Vector Length} \times \text{Bytes per Element}\end{aligned} \tag{6}$$

If we consider the number of bytes per element to be 4, then the memory requirement would be around 302MB. The entire feature database can be stored on 302MB of memory without affecting the retrieval performance. Moreover, the length of the features vector also reduces retrieval time. Similarity-based approaches have a trade-off between several features and retrieval time.

4.7 Indexing and retrieval of extracted features

After fine-tuning the neural network and obtaining feature vectors for all dataset images, the next step is to index these feature vectors for efficient retrieval. The goal is to organize the features to minimize the time required to match query features with the database and retrieve the most similar indices using a search algorithm. Efficient indexing is crucial for optimizing the performance of the retrieval system.

The proposed work uses the approximate nearest neighbor search technique, i.e., Approximate Nearest Neighbors Oh Yeah(Annoy) [40]. Annoy takes a set of feature vectors as input, where each vector represents a data point in a high-dimensional space. It constructs a tree-based data structure known as a random projection tree. Random projections is a technique used in high-dimensional data to reduce the dimensionality of the data while approximately preserving the pairwise distances between points. The idea is to project the original high-dimensional points onto a lower-dimensional subspace so that the pairwise distances between the projected points are close to the original distances. Annoy creates multiple random projections of the feature vectors and organizes them into a tree structure to form the index. When given a query point (a feature vector), this algorithm traverses the random projection tree to find the approximate nearest neighbors of the query point. It performs a limited search in the tree to identify the closest data points, likely to be the nearest neighbors. The number of building trees depends on the size of the dataset, which is usually the square root of the dataset size. However, the greater number of trees leads to a longer retrieval time.

Annoy measures the approximations, which means it provides an efficient way to find good approximations. The trade-off is that it sacrifices some accuracy for faster retrieval times, making it suitable for large-scale datasets. It is an indexing method that excels in high-dimensional data and closest neighbor search. It can handle large datasets with millions of points, making it ideal for applications requiring vast high-dimensional data searches. The indexing structure is small, requiring no memory load, and the number of trees can be adjusted for accuracy and performance. In contrast to KD trees and Ball Trees, ANNOY's use of random projection trees effectively partitions high-dimensional environments. The indexing structure is small, and there is no need to load the complete dataset into memory. Annoy can perform queries quickly once the index is generated, in less than a second for huge datasets, many times.

5 Evaluation metrics

To assess an image retrieval system, it's crucial to consider the specific task and requirements to select the suitable evaluation metric(s) for an accurate assessment [41]. Precision and recall are used to assess the performance of this model, which are commonly used metrics for image retrieval. Table 4 summarises the values of precision and

Table 4 Values of evaluation metrics for retrieved images using proposed framework

Metric	Flower photos	Caltech-101	Book covers	Caltech-256
Precision	0.2 to 1	0.2 to 1	0.2 to 1	0.2 to 1
Recall	1	1	1	0.9991
Accuracy	1	1	1	1
F-measure	0.33 to 1	0.33 to 1	0.33 to 1	0.33 to 1

recall for different query images for each dataset using the proposed framework.

- **Precision**: It measures the proportion of relevant images among the top-K retrieved images for each query. P@K evaluates the system's precision when considering only the top-K retrieved images.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

- **Recall**: This metric quantifies the fraction of relevant images retrieved within the top-K retrieved images. R@K assesses the system's effectiveness in retrieving relevant images within the top-K results.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

- **Accuracy**: Accuracy measures how well the system retrieves relevant images in response to a query. It is often calculated as the ratio of the number of correctly retrieved relevant images to the total number of relevant images. The accuracy metric helps assess the effectiveness of the CBIR system in returning images. The accuracy formula for CBIR can be expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

- **F-Measure**: F-Measure, or F1 Score, is a measure of a test's accuracy, combining both precision and recall. F-Measure is the harmonic mean of precision and recall. A perfect system would have both precision and recall equal to 1, resulting in an F-measure of 1. A lower F-measure indicates a trade-off between precision and recall. The following expression defines it:

$$\text{F} - \text{measure} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

6 Experiment and results

6.1 Datasets

The three datasets Caltech-101 [16], flower photos [17] and book covers [18] datasets taken from Kaggle been tested for their performance on the proposed framework. 45,416 images are divided into 140 different categories. During training, we have split the dataset images and labels into test-train categories with a test size of 0.3. The selection of the dataset is crucial for the evaluation of automatic feature extraction techniques. A few of the used images are presented in Fig. 7.

6.1.1 Flower photos

Flower photos [17] dataset contains a total of 3670 images divided into 5 categories of different types of flowers, namely daisies, dandelions, roses, sunflowers, and tulips. The images of this dataset have been resized to (224,224,3), and resized images are supplied as input to the model. This dataset is chosen to include a diversity of visual features such as color, texture, and shapes to test the robustness of the proposed framework against real-world applications.

6.1.2 Caltech-101

Caltech-101 [16] dataset consists of images categorized into 101 different classes, each representing a distinct object category. Within each class, there are approximately 40 to 800 images, with an average of around 50 images per class. The images in the dataset have a resolution of approximately 300× 200 pixels. The images of this dataset were also resized to (224,224,3), and resized images were supplied as input to the model. This dataset poses the challenge of intra-class variability. This helps assess the generalization capability of the feature extraction technique. This dataset also sets a benchmark of comparison with the existing techniques.

6.1.3 Book covers datasets

Book covers [18] dataset contains 33 classes (book categories), and each category contains close to 1000 images. All the images are in jpg format, and there are 32,581 images in total. All the images were resized to (128,128,3). This dataset is included for the high variability in design that involves a wide range of designs such as fonts, images, and layouts to test the adaptability of the proposed framework.

Fig. 7 Sample images from the dataset

6.2 Experimentation

The new model was used to predict the features of the query image and all the images in the dataset and create a feature database. The features from this database are indexed using ANNOY [40] indexing. Euclidean similarity [42] of ANNOY algorithm is used to find the similarity of features of query image and and database images. The retrieval time and matching of similar features also depend on the number of trees used for indexing the features. In this case, the optimal value of the number of building trees for the complete dataset is 200, approximately the square root of the total number of images in the database. The time taken for retrieval is directly proportional to the number of building trees (Fig. 8; Tables 5, 6, and 7).

6.3 Results and analysis

To evaluate the system's performance, 100 query images were selected from the dataset. The corresponding top 5 images are retrieved entirely based on the content for each query image. For the retrieved images, all four metrics have been computed as mentioned in Sect. 5. Precision, recall, accuracy, and F-measure were calculated using Eqs. 7, 8, 9, and 10 respectively. To calculate all the metrics, the numerical labels have been assigned to all the image categories in both the database and the query set. Numerical labels of retrieved images are compared to those of a query image, and if found to be similar, they are relevant retrievals. The results for query images with 100% recall and accuracy are presented in Table 4 (Tables 8 and 9).

The proposed framework shows varying precision values ranging from 0.2 to 1.0 for the entire database. Since precision measures the accuracy of the retrieved images, a higher precision signifies an increased ratio of relevant images within the retrieved set. The minimum value of 0.2 indicates that at least one relevant image was retrieved among the 5 retrieved images, and precision is not at all 0 for a single query image in the database. In this case,

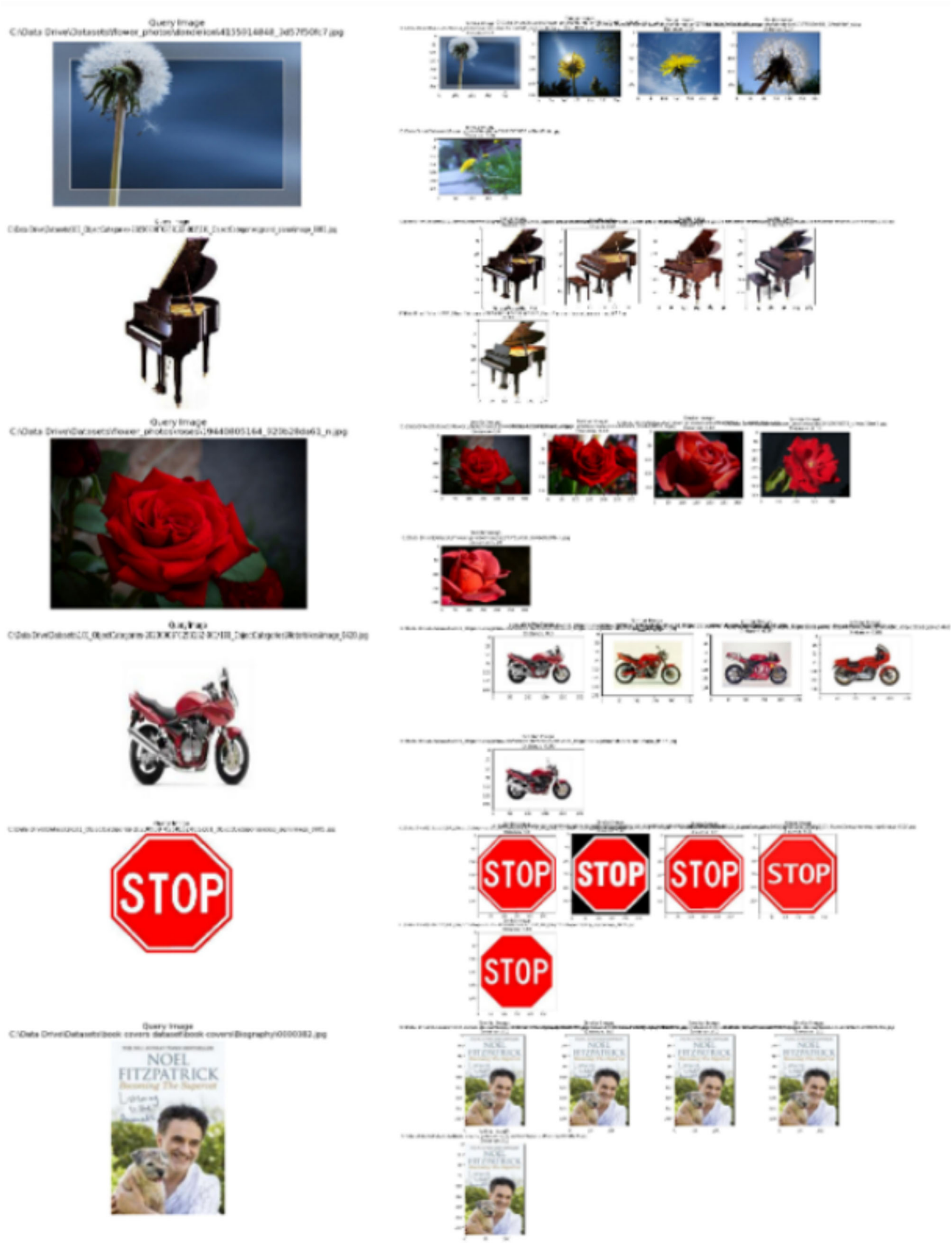


Fig. 8 Given query images and corresponding retrieved images using proposed framework

Table 5 Average retrieval time using proposed framework

	Flower photos	Caltech-101	Book covers	Catech-256
Average Retrieval Time(ms)	2.01	3.31	5.78	614.00

Table 6 Recall and accuracy of proposed framework

	Flower photos	Caltech-101	Book covers	Caltech-256
Recall	1	1	1	99.91
Accuracy	1	1	1	1

Table 7 Comparison of average retrieval time (ms)

Dataset	Subhadip [11]	Proposed framework
Caltech-101	7.9	3.31

Table 8 Comparative analysis of existing and proposed framework

Model	Dataset type	Recall	F-score	Accuracy
VGG-16 [4]	Categorical	0.037	0.91	–
VGG-19 [4]	Categorical	0.043	0.94	–
Ahmed Ali [13]	Categorical	0.0955	–	–
Ouni *et.al* [43]	Miscellaneous	0.62	0.52	0.91
Proposed framework	Categorical	1	0.70	1

precision tends to be relatively high, reaching 1.0 in many cases, suggesting that the system retrieves a significant number of relevant images. The recall values are consistently 1.0 across all queries. AS recall measures the ability of the system to retrieve all relevant images, a recall of 1.0 indicates that the system retrieves all relevant images for each query image.

While high recall is desirable, it is essential to balance it with precision. Like recall the accuracy values are also consistently 1.0. Because accuracy measures the overall correctness of the system's retrieved results, a value of 1.0 suggests that all retrieved images are relevant and correctly retrieved. While high accuracy is desirable, other metrics have been considered, like precision and recall, to have a comprehensive evaluation. To verify this outcome of recall and accuracy being consistently high, the proposed framework is tested randomly on the images taken from the internet that do not belong to our dataset. Our model has performed exceptionally well on unseen data and has maintained the same for all categories. F-measure, a harmonic mean of precision and recall, ranges from 0.33 to 1.0. F-measure offers a balanced evaluation, taking into account both precision and recall. A higher F-measure indicates a better balance between precision and recall. The system achieves relatively high F-measure values, suggesting a good balance between precision and recall. The system shows promising performance, with high precision, recall, accuracy, and F-measure values. The consistently high recall values may indicate that the system successfully retrieves all relevant images. The high precision values suggest that the retrieved images are mostly relevant, contributing to the overall accuracy of the system.

6.4 Caltech-256 dataset

To further illustrate the performance analysis of the proposed approach, it was tested on the Caltech-256 dataset [47]. Each photograph in this collection has dimensions of 300×200. There are 256 picture semantic categories, each with at least 80 images. This collection contains 30,607 photos. The training and testing sets for this dataset were divided in the ratio of 70:30. The suggested system was tested on the Caltech-256 dataset with 2250 query photos and yielded outstanding results. The framework had a mean Average Precision (mAP) of 69.91%, an average Recall (mAR) of 99.91%, and an F1 score of 77.80%. The average retrieval time (ART) per query was 614.00 milliseconds, with a total accuracy of one.

When compared to existing methodologies, the suggested framework performs much better. Shrif (47.52%) Lorenzo (18.17%), and Yousuf (30.3%) all have significantly lower mAPs than 69.9%. Similarly, the 99.9% recall rate is a significant improvement above Shrif et al. (9.5%) and Yousuf et al. (6.6%). Lorenzo et al. did not offer recall data for comparison.

This improved result demonstrates that the suggested framework is extremely effective in accurately retrieving images from the Caltech-256 dataset, demonstrating its durability and reliability when compared to other cutting-edge methodologies. The excellent precision and recall indicate that the framework is accurate and comprehensive in retrieving relevant photos.

Table 9 Comparison of proposed framework with existing techniques using mAP and mAR for Caltech-256 dataset

	Shrif [44]	Lorenzo [45]	Yousuf [46]	Proposed framework
mAP	47.52	18.17	30.3	69.9
mAR	9.5	-	6.6	99.9

6.5 Average retrieval time

In CBIR, the average retrieval time is a crucial measure for assessing the practical usability and real-time performance of a CBIR system. It is an important efficiency measure that quantifies the time it takes for the system to retrieve relevant images in response to a query. The average retrieval time has been measured in milliseconds on non-GPU machines. ANNOY, a tree-based structure, indexes extracted features since it can quickly prune the search space, making it suitable for real-time applications. For our work, the average retrieval time is mentioned in Table 5. Figure 9 shows a comprehensive comparison of the retrieval time of the proposed framework and the existing technique on a local machine with the following Configuration:

Hardware requirements:

- 12th Gen Intel(R) Core(TM) i7-12700 H 2.30 GHz
- 16.0 GB (15.6 GB usable)

Software requirements:

- Python Version: 3.11.4,
- TensorFlow Version: 2.16.2
- Keras Version: 3.4.1
- PIL (Pillow) Version: 9.4.0
- NumPy Version: 1.24.3

6.6 Discussion

The proposed image retrieval framework is very sensitive to the characteristics of the query images and effective in finding relevant matches. The algorithm is well-tuned to the features of the query images, leading to the achievement of high recall. For poorly performed categories where the precision and recall range from 0.2 to 0.8, the retrieved images are different in terms of labels but these are similar in terms of content. Since precision is the measure of the portion of images among the retrieved images that are similar to the query image, in the worst-case scenario, the system is capable of retrieving at least one similar image among the top five images while others are similar in terms of content but belong to different categories leading to the low value of recall. The retrieval results with varying precision from 0.2 to 0.8 are shown in Fig. 10. The visual

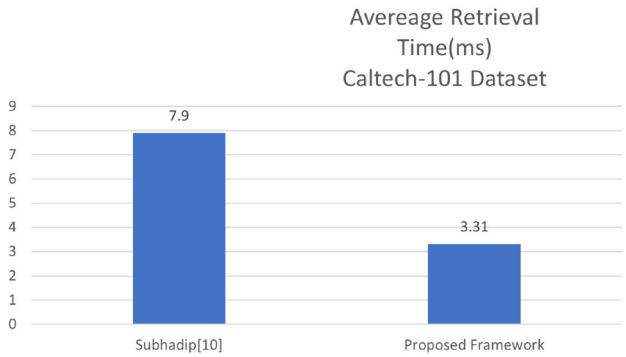


Fig. 9 Comparison of average retrieval time of [11] and proposed framework for Caltech-101 dataset

results prove the sensitivity of the proposed technique to the image features. To extend the evaluation one step further, Fig. 11 presents the performance of the system on random images from the internet. The values of all the evaluation metrics vary on unseen data in a similar manner as that of data used for training purposes. The system shows promising performance, with high precision, recall, accuracy, and F-measure values. Therefore, the framework's ability to distinguish target features justifies the high values of recall and accuracy.

Our research involves a comprehensive comparison with the work presented by Sikandar [10]. In Sikandar's approach, concatenated features from ResNet50 and VGG16 were employed, resulting in a final feature vector of length 2560. While Sikandar achieved a feature vector length of 2560 through concatenation, the proposed approach has achieved a comparable and potentially more compact feature vector. Using intermediate layers and a modified architecture contributes to a more fine-tuned and concise representation of image features. Our comparison has extended beyond feature vector length and includes an in-depth analysis of both performance metrics and model complexity. For the Caltech-101 dataset, the performance comparison with existing technique [33] is given in Fig. 12. The existing technique has obtained mAP of 98.5%, mAR of 99%, F1 of 98%, and accuracy of 98.2%, respectively. In contrast, our approach is performing better in terms of all the performance metrics with values of 98.7% for mAP while the perfect value of 1 for mAR, average F1, and accuracy for the Caltech-101 dataset.

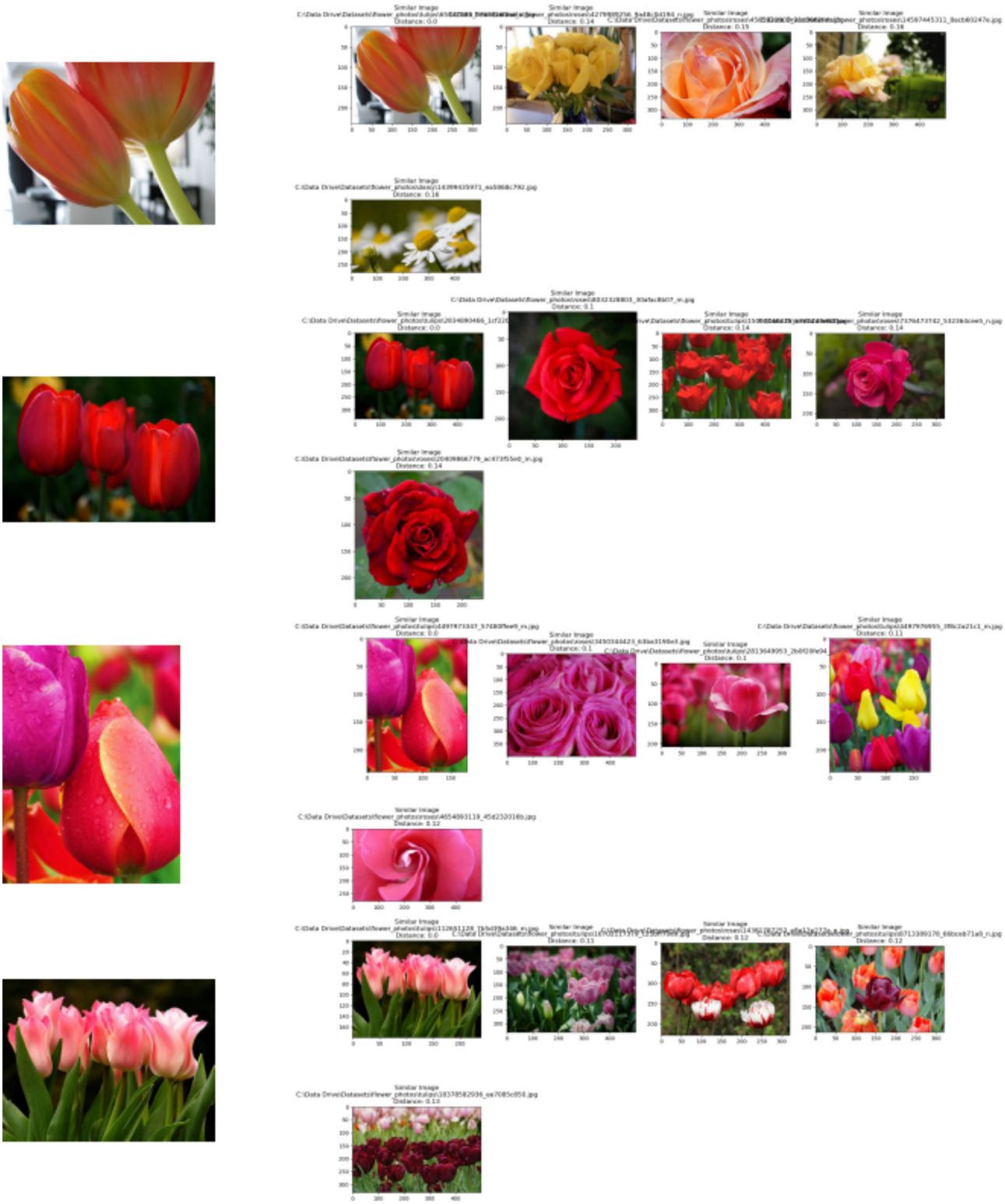


Fig. 10 Sample results using proposed framework with varying precision values of 0.2, 0.4, 0.6 and 0.8 respectively but high recall and accuracy

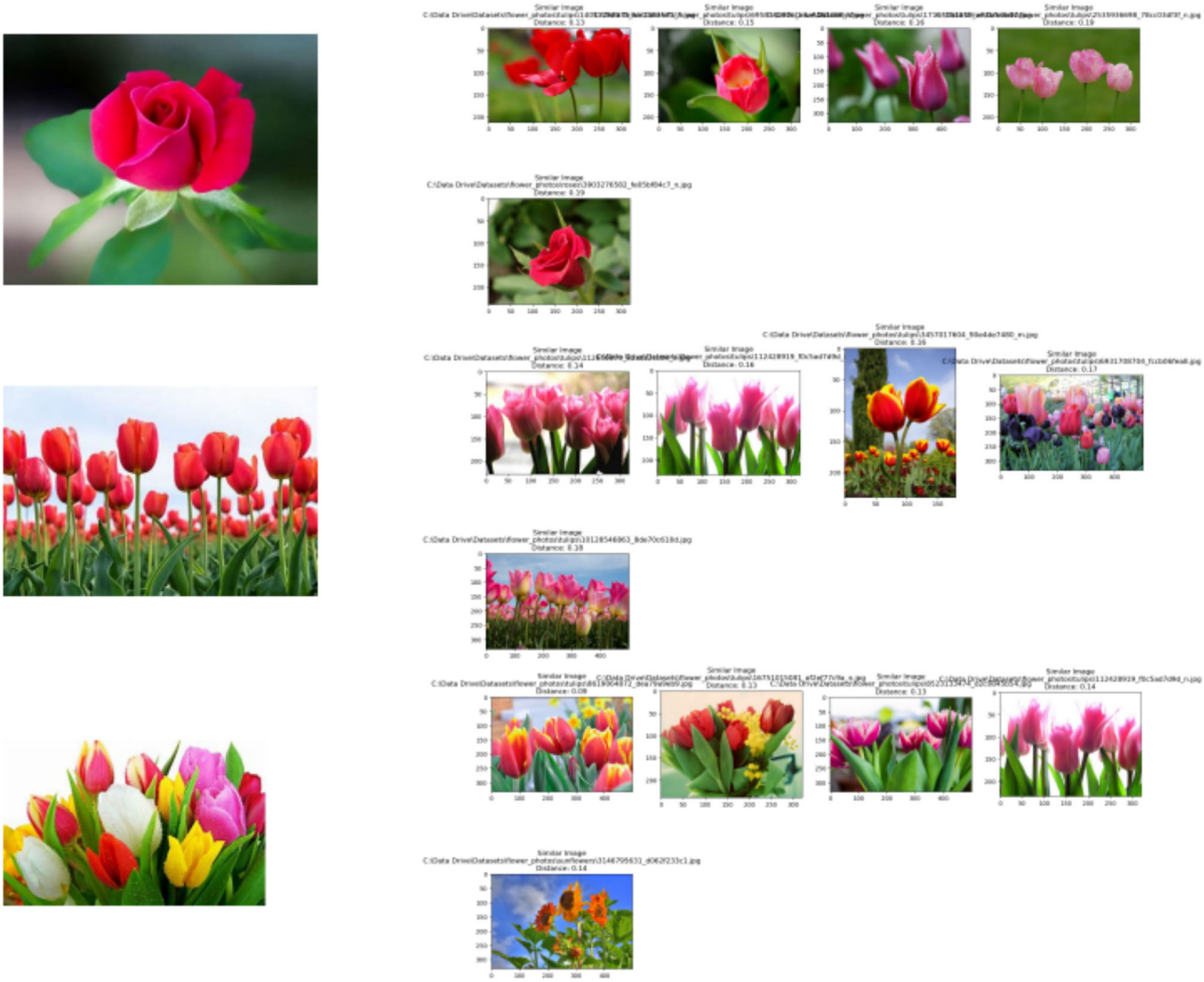


Fig. 11 Retrieval results using proposed framework on unseen images (query images taken randomly from Internet

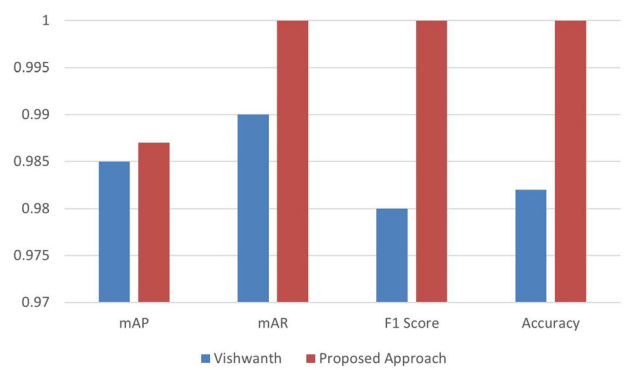


Fig. 12 Performance comparison of proposed framework with existing technique [33] for Caltech-101 dataset

7 Ablation study

In this part of the paper, ablation research is conducted to examine how different changes to the baseline model affect the model's performance. The baseline is the original model, which is based on the EfficientNetB0 architecture. Several modifications are investigated by methodically eliminating or changing particular elements, such as the top layer and L2 normalization layer, as well as by modifying the input shape and embedding size.

7.1 Experimental setup

Baseline Model: The original model has about 5.3 million parameters and uses the input shape (224, 224, 3). The number of hyper-parameters, file size, length of final feature vector, and precision are used as metrics. Different variants of the baseline model are explored based on the

input of sizes (128, 128, 3) and (224, 224, 3) and different output shapes of the order of 128,256 and 512 as depicted below:

- Original Model: The original model accepted the input shape of (224, 224, 3) and has 5.3M parameters and a model size of 29MB.
- Model without Top Layer: The input shape was reduced to (128, 128, 3), yielding an output feature vector with the shape of (4, 4, 1280) and 4 million parameters.
- Modified model without top Layer and added L2 Normalization Layer (embedding size = 128 and input shape = (128, 128, 3)): resulting in 4.24 million parameters. The final output feature vector of the modified model is 1616.
- Modified model without top layer and added L2 Normalization Layer (embedding size = 128 and input shape = (224, 224, 3)): This preserves 4.24 million parameters. The final output feature vector of the modified model is 1616.
- Modified model with L2 Normalization Layer (embedding size = 256 and input shape = (128, 128, 3)): This model has an output vector of length 1744 with 4.25 million parameters.
- Modified model with L2 Normalization Layer (embedding size = 256 and input shape = (224, 224, 3)): This model also has an output vector of length 1744 with 4.25 million parameters.
- Modified model with L2 Normalization Layer (embedding size = 512 and input shape = (224, 224, 3)): this model yields 4.29 million parameters and an output vector of length 2000.

7.2 Results and analysis

- Model without Top Layer: The number of parameters and model are greatly reduced to 4 M and 15.45 MB by eliminating the last layers of the model. However, the produced feature vector was a 4D tensor (4, 4, 1280), which might make feature matching more difficult. Since this approach could not offer precise performance metrics, more research in a similar setting is necessary.
- Modified model eliminating last layers and added separable layer and L2 normalization layer (Embedding Size = 128):
  - The model contained 4.24M parameters and a feature vector of length 1616 with the input shape of (128, 128, 3). With a total size of 16.18 MB, this model obtained precision values within the range of 0 to 0.4.
  - For the same embedding size of 128 and input size of (224,224,3), all the metrics hold almost similar values, i.e., 4.24M, 1616, and 16.18MB, as mentioned above. Further, there is not much improvement in precision.
- Modified model eliminating last layers and added separable layer and L2 normalization layer (Embedding Size = 256)
  - With input shape (128, 128, 3), the model had 4.25M parameters and the feature vector length of 1744, a model size of 16.24MB, and precision ranging between 0 to 0.8 for all the query images.
  - For the same embedding size of 256 and input size of (224,224,3), all the metrics hold almost similar values, i.e., 4.25M, 1744, and 16.24MB, as mentioned above. Precision, in this case, ranges between 0.2 to 1.
- Modified model eliminating last layers and added separable layer and L2 normalization layer (Embedding Size = 512)
  - The input shape of (224, 224, 3) produced 4.29M parameters and a feature vector length of 2000 for the largest embedding size of 512. With a model size of 16.37 MB, this configuration yielded a precision range of 0.6 to 1. The findings suggest that improving performance may always result from larger embedding sizes of the output layer and can capture more discriminative features.

7.3 Key findings

- Increased Input Size: The trade-off between input size and model efficiency is highlighted by the fact that increasing the embedding size from 128 to 224 improved precision but does not always result in higher performance.
- Embedding Size: Performance is usually better across models when the output embedding size is increased from 128 to 256 and 512. This suggests that higher-resolution inputs facilitate better feature extraction.
- Model Modification: Removing the last softmax layer and reducing the model complexity by adding separable layers and an L2 normalization layer still resulted in competitive performance, especially with the optimal input size and embedding size configurations.
- Optimizer Parameters: Since the feature extractor model is composed of the intermediate layers and final layer, the optimizer parameters (8.399M and 32.04MB) increase during the fine-tuning of the modified model; however, the feature extractor model's parameters stay at 4.2M for all input and embedding size variations. As

input form and embedding size grow, there is a modest increase in optimizer parameters.

This ablation study demonstrates that careful consideration of input size, embedding size, and model complexity can lead to more efficient models without significantly compromising performance. These findings will guide future optimizations and potential simplifications of the proposed model for different applications.

8 Conclusion and future scope

This study proves that the intermediate layers of pre-trained models can produce the most robust and distinctive features of optimal dimension. The proposed model can retrieve all the relevant images with perfect recall and accuracy of 1 for the complete dataset, along with significant values of precision and F-measure. The base of the proposed model is the recent EfficientNetB0 pre-trained model. An effort has been made to reduce the complexity and number of hyperparameters by adding separable CNN on the top of EfficientNetB0, which is 4,258,083 only, and the length of the feature vector is 1744, which sufficiently signifies the extraction of identified features. This modified model works in three parts: feature extraction, indexing, and retrieval. The modified model is used as a feature extractor from the intermediate layers and obtains a feature vector of lower dimensions. The features obtained using this feature extractor are indexed using ANNOY indexing. The matching process between the features of the query image and those of the database images is executed through Euclidean similarity. This indexing method significantly reduces the retrieval time. Recent CBIR models tend to work on images without a label. Given the time-consuming nature of image labeling or annotation for huge datasets, alternative approaches are sought to expedite the process. In context to this, unsupervised methods need to be explored. In the future, we envision developing an unsupervised image retrieval framework. By incorporating unsupervised learning techniques, our goal is to develop a robust and scalable CBIR system that can effectively retrieve relevant images without the need for manual annotation. This approach aims to streamline the image retrieval process and enhance the system's adaptability to diverse and large-scale image datasets.

Author contributions Sunita Rani, Shalini Batra, and Geeta Kasana conceptualized and designed the study. Sunita Rani conducted the literature review and performed data collection and analysis. Sunita Rani and Geeta Kasana contributed to the development of the methodology and conducted experimentation. Shalini Batra drafted the initial manuscript and provided critical revisions and feedback. All the authors reviewed and approved the final version of the manuscript for submission.

Funding This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interest The authors declare no competing interests.

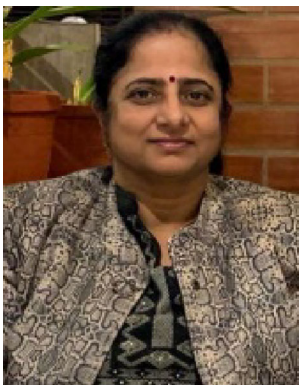
References

1. Deselaers, T., Keysers, D., Ney, H.: Features for image retrieval: an experimental comparison. Inf. Retr. **11**, 77–107 (2008). https://doi.org/10.1007/s10791-007-9039-3
2. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 91–110 (2004)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. IEEE Conf. Comput. Vis. Pattern Recogn.(CVPR) **2016**, 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
4. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014). https://api.semanticscholar.org/CorpusID:14124313
5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. CoRR abs/1610.02357 (2016). arXiv:1610.02357
6. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L.: Imagenet large scale visual recognition challenge (2014). arXiv:1409.0575
7. Arun, S.D.M.K.K.S., Govindan, V.K.: Enhanced bag of visual words representations for content based image retrieval: a comparative study. Artif. Intell. Rev. **53**, 1615–1653 (2020)
8. Jian Zhang, Q.W., Cao, Y.: Vector of locally and adaptively aggregated descriptors for image feature representation. Pattern Recogn. **116**, 107952 (2021). https://doi.org/10.1016/j.patcog.2021.107952
9. Abdel-Nabi, H., Al-Naymat, G., Awajan, A.: Content based image retrieval approach using deep learning, in: 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), pp. 1–8 (2019). https://doi.org/10.1109/ICTCS.2019.8923042
10. Sikandar, S., Mahum, R., Alsalman, A.: A novel hybrid approach for a content-based image retrieval using feature fusion, Appl. Sci. **13** (7) (2023). https://doi.org/10.3390/app13074581. https://www.mdpi.com/2076-3417/13/7/4581
11. Maji, S., Bose, S.: CBIR using features derived by deep learning. ACM/IMS Trans. Data Sci. **2**(3), 1–24 (2021). https://doi.org/10.1145/3470568
12. Alappat, A. L., Nakhate, P., Suman, S., Chandurkar, A., Pimpalkhute, V., Jain, T.: Cbir using pre-trained neural networks (2021). arXiv:2110.14455
13. Ahmed, A.: Pre-trained CNNs models for content based image retrieval. Int. J. Adv. Comput. Sci. Appl. **12**, 2021 (2021). https://doi.org/10.14569/IJACSA.2021.0120723
14. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval (2014). arXiv:1404.1777
15. Chen, W., Liu, Y., Wang, W., Bakker, E., Georgiou, T., Fieguth, P., Liu, L., Lew, M.: Deep image retrieval: A survey. 2101.11282 (2021) 1–21. https://arxiv.org/abs/2101.11282

16. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. Comput. Vis. Image Understand. **106**(1), 59–70 (2007)
17. Team, T.: Flower photos dataset. Licensed under the creative commons by-attribution license (CC BY 2.0) (2019). https://creativecommons.org/licenses/by/2.0/
18. Iwana, B. K., Raza Rizvi, S. T., Ahmed, S., Dengel, A., Uchida, S.: Judging a book by its cover. arXiv preprint arXiv:1610.09204 (2016)
19. Khan, A., Sohail, A., Zahoora, U., Qureshi, A.S.: A survey of the recent architectures of deep convolutional neural networks. Artif. Intell. Rev. **53**(8), 5455–5516 (2020). https://doi.org/10.1007/s10462-020-09825-6
20. Razavian, A. S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. CoRR abs/1403.6382 (2014). arXiv:1403.6382
21. Dubey, S.R.: A decade survey of content based image retrieval using deep learning. IEEE Trans. Circ. Syst. Video Technol. **32**(5), 2687–2704 (2022). https://doi.org/10.1109/TCSVT.2021.3080920
22. Wan, J., Wang, D., Hoi, S. C. H., Wu, P., Zhu, J., Zhang, Y., Li, J.: Deep learning for content-based image retrieval: A comprehensive study, in: Proceedings of the 22nd ACM international conference on multimedia, MM '14. Association for Computing Machinery, New York, pp. 157–166 (2014). https://doi.org/10.1145/2647868.2654948
23. Abdel-Nabi, H., Al-Naymat, G., Awajan, A.: Content based image retrieval approach using deep learning, in: 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), pp. 1–8 (2019). https://doi.org/10.1109/ICTCS.2019.8923042
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM (2017). https://doi.org/10.1145/3065386
25. Gao Huang, Z. L., van der Maaten, L., Weinberger, K. Q.: Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708 (2017)
26. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-ResNet and the impact of residual connections on learning. AAAI Press (2017)
27. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510–4520 (2018)
28. Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V.: Learning transferable architectures for scalable image recognition (2018). arXiv:1707.07012
29. Szegedy, C., Vanhoucke, V., Sergey Ioffe, J. S., Wojna, Z.: Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567 (2015)
30. Kumar, S., Singh, M.K., Mishra, M.K.: Improve content-based image retrieval using deep learning model. J. Phys.: Conf. Ser. **2327**(1), 012028 (2022). https://doi.org/10.1088/1742-6596/2327/1/012028
31. Salih, S., Abdulla, A.: An effective bi-layer content-based image retrieval technique. J. Supercomput. (2022). https://doi.org/10.1007/s11227-022-04748-1
32. Salih, F., Abdulla, A.: Two-layer content-based image retrieval technique for improving effectiveness. Multimedia Tools Appl. (2023). https://doi.org/10.1007/s11042-023-14678-6
33. Mahalle, V., Kandoi, N., Patil, S.: A powerful method for interactive content-based image retrieval by variable compressed convolutional info neural networks. Visual Comput (2023). https://doi.org/10.1007/s00371-023-03104-5
34. Shang, J.-J., Phipps, N., Wey, I.-C., Teo, T.H.: A-DSCNN: depthwise separable convolutional neural network inference chip design using an approximate multiplier. Chips **2**(3), 159–172 (2023). https://doi.org/10.3390/chips2030010
35. Tan, M., Le, Q. V.: Efficientnet: rethinking model scaling for convolutional neural networks. CoRR abs/1905.11946 (2019). arXiv:1905.11946
36. Goodfellow, I., Bengio, Y., Courville, A.: Efficient convolution algorithms. In: Goodfellow, I., Bengio, Y., Courville, A. (eds.) Deep learning, 1st edn., pp. 350–354. MIT Press (2016)
37. Hua, B., Tran, M., Yeung, S.: Point-wise convolutional neural network. CoRR abs/1712.05245 (2017). arXiv:1712.05245
38. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: efficient convolutional neural networks for mobile vision applications. CoRR abs/1704.04861 (2017). arXiv:1704.04861
39. Sarkar, D.: A comprehensive hands-on guide to transfer learning with real-world applications in deep learning, deep learning on steroids with the power of knowledge transfer! (2018). https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a
40. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 2227–2240 (2014). https://doi.org/10.1109/TPAMI.2014.2321376
41. Arora, M., Kanjilal, U., Varshney, D.: Evaluation of information retrieval: precision and recall. Int. J. Indian Cult. Bus. Manag. **12**, 224 (2016). https://doi.org/10.1504/IJICBM.2016.074482
42. Elmore, K.L., Richman, M.B.: Euclidean distance as a similarity metric for principal component analysis. Monthly Weather Rev **129**(3), 540–549 (2001)
43. Ouni, A., Royer, E., Chevaldonné, M., Dhome, M.: Leveraging semantic segmentation for hybrid image retrieval methods. Neural Comput. Appl. (2021). https://doi.org/10.1007/s00521-021-06087-3
44. Sharif, U., Mehmood, Z., Mahmood, T., Javid, M. A., Rehman, A., Saba, T.: Scene analysis and search using local features and support vector machine for effective content-based image retrieval. Artif. Intell. Rev. 1–25 (2018). https://api.semanticscholar.org/CorpusID:64594474
45. Putzu, L., Piras, L., Giacinto, G.: Convolutional neural networks for relevance feedback in content based image retrieval: a content based image retrieval system that exploits convolutional neural networks both for feature extraction and for relevance feedback. Multimedia Tools Appl. **79**(37–38), 26995–27021 (2020). https://doi.org/10.1007/s11042-020-09292-9
46. Yousuf, M., Mehmood, Z., Habib, H.A., Mahmood, T., Saba, T., Rehman, A., Rashid, M.: A novel technique based on visual words fusion analysis of sparse features for effective content-based image retrieval. Math. Probl. Eng. **2018**, 13 (2018). https://doi.org/10.1155/2018/2134395
47. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007). https://api.semanticscholar.org/CorpusID:118828957

Sunita Rani is currently pursuing their Ph.D. in Computer Science and Engineering Department at Thapar Institute of Engineering and Technology, Patiala, with a research focus on image retrieval and image processing. Her work primarily involves developing novel approaches to image retrieval using deep learning architectures and optimizing image retrieval models for large-scale data analysis. Sunita Rani has previously completed her Master's Degree in Computer Science and Engineering from Department of Computer Science, Punjabi University, Patiala. She is passionate about advancing computer vision tasks and aims to contribute to solving real-world challenges by developing scalable and efficient algorithms that improve image recognition, retrieval, and analysis systems for various practical applications.

Geeta Kasana is working as Assistant Professor in Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, India. She earned her Ph.D. degree in Information Security from Thapar University. Her research interests encompass image processing and information security. With around 15 years of teaching and research experience, she has published 30 research papers in esteemed international journals and conferences. She is guiding Ph.D. students in the area of information security, information retrieval and deep learning.



Shalini Batra is a Professor and Head of the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, joined the institute in 2002 and has served in various administrative positions. She has guided more than 50 M.E. scholars and seven Ph.D. scholars. She has more than 100 publications in Journals of repute and International Conferences. Her major research areas are Big Data Analytics, with a special focus on Probabilistic Data Structures, Machine Learning, and Image processing.