

Enhancing Content Based Image Retrieval Through Deep Convolutional Feature Extraction Combined with Ensemble Learning Classifiers

Dr. Kishor Rajendrakumar Shinde¹, Dr. Alok Agarwal²

¹ Assistant Professor, Electronics and Telecommunication Engineering, Yashoda Technical Campus, Satara, Maharashtra, India,
Email: kshinde3@gmail.com

² Assistant Professor, Department of Computer Science and Engineering, South Asian University, New Delhi, India,
Email: alokagarwal26aaa@gmail.com

Abstract: Searching a large image database by keywords alone rarely gives satisfactory results, mainly because a keyword can never fully describe what is actually inside a picture. This gap between what a machine reads as metadata and what a human sees is usually called the semantic gap, and Content Based Image Retrieval, or CBIR, exists to close it. This paper builds on an earlier VGG16 and XGBoost based CBIR pipeline and extends it by placing the ensemble classification stage at the centre of the study. VGG16 is used purely as a feature extractor, and the resulting feature vectors are passed to a gradient boosted ensemble for prediction and similarity ranking. The system was implemented and tested on 1000 images drawn from the public Kaggle Flowers Recognition dataset, across five categories, daisy, dandelion, rose, sunflower and tulip. The trained model reached a training accuracy of 1.00 but a validation accuracy of only 0.72, a gap pointing to overfitting on the high dimensional 25088 length feature vectors relative to the dataset size, discussed openly here as one of the more useful findings of the study. The paper reviews thirty related works and closes with a comparison table and directions for future work.

Keywords: Content Based Image Retrieval, Deep Learning, VGG16, Ensemble Learning, XGBoost, Feature Extraction, Gradient Boosting

1. Introduction

The amount of image data generated every day keeps growing, and it is growing faster than any manual tagging system can keep up with. Cameras in phones, medical scanners, satellites and industrial sensors all add to this pile, and at some point searching through it by typing keywords stops being practical. Content Based Image Retrieval was developed to deal with exactly this problem. Instead of relying on a caption or a filename, a CBIR system looks at the actual pixels, colours, edges and textures of an image and uses that visual content to find similar images in a database [1], [2].

A typical CBIR system works in two stages. First, during indexing, low level features are pulled out of every image in the database and stored as feature vectors. Second, during retrieval, the same kind of features are extracted from a query image submitted by the user, and these are matched against the stored vectors using some distance or similarity measure [3]. For years this feature extraction step relied on handcrafted descriptors such as colour histograms, edge maps or texture matrices. These descriptors work, but they struggle whenever an image contains complex or overlapping visual patterns, and this is really where the semantic gap becomes visible.

Deep learning changed this picture considerably. Convolutional Neural Networks, or CNNs, learn features directly from the data rather than depending on a person to define what to measure, and this has pushed CBIR accuracy up by a wide margin [21]. Among the many CNN architectures available, VGG16 remains a popular choice for feature



extraction because its layered structure captures both fine detail and high level shape information without being overly complicated to train or reuse [26].

Even with strong features, the classification or ranking stage still decides how well a system actually performs. This paper focuses on that second stage. Rather than using a single classifier as a formality after feature extraction, the ensemble learning step built on XGBoost is treated as the core contribution here, since it is the boosting process that turns a set of raw deep features into a confident, well separated prediction. Section III explains the architecture, Section IV covers the experimental setup, and Section V presents the results obtained on a 1000 image, five category flower dataset.

2. Literature Review

Early CBIR research leaned heavily on handcrafted, low level descriptors. The Color Co-occurrence Matrix has been one of the most widely used tools for this purpose and appears across a large number of CBIR studies [5], [6], [7]. Shim and Choi [5] proposed a modified colour co-occurrence matrix for indexing, Huang [6] worked on colour spatial indexing for large image collections, and Ojala et al. [7] used HSV correlograms for semantic retrieval, showing that colour correlation across neighbouring pixels carries useful information beyond a plain histogram.

Several works combined more than one type of descriptor to improve accuracy. Bose et al. [8] applied discriminant analysis to strengthen retrieval performance, and Lohite and Pawar [9] paired local features with an SVM classifier, an early attempt to bring a machine learning classifier into the pipeline rather than relying only on distance measures. Mehmood [10] combined local and global histograms of visual words, an idea pushed further by Yousuf et al. [11] through fused sparse visual word features, and by Sharif et al. [12], who used local features with an SVM for scene analysis and search.

Texture and shape based descriptors also received considerable attention. Ashraf et al. [13] used embedded neural networks with bandletized regions, Obulesu et al. [14] proposed a multi motif co-occurrence matrix, and Khokhar and Verma [15] combined multiple features with a backpropagation neural network. Liu et al. [16] surveyed high level semantic based retrieval methods, useful background for why the semantic gap remains persistent. Safar et al. [17] compared shape based retrieval methods, Manjunath and Ma [18] studied texture features for browsing, and Kokare et al. [19] used rotated wavelet filters for texture retrieval.

Ghughe and colleagues contributed several textural and transform based approaches, including a textural feature based CBIR method [20], a Radon projection based approach [22], and a multi resolution feature study [25]. Sastry et al. [21] proposed a modified Gabor function for retrieval, Neumann and Gegenfurtner [23] examined perceptual similarity, and Suhasini et al. [24] used colour histogram processing, all reinforcing how central colour and texture cues have been to this field.

More recently, deep learning has taken over much of the feature extraction workload. Jaroslavceva [26] studied image retrieval using CNNs in TensorFlow, and Gautama and Khann [27] reviewed CNN based deep learning models applied to CBIR. Hameed et al. [28] published a broad review of recent CBIR trends, helping place CNN based methods against older handcrafted methods. Cha [2] and Meharban and Priya [3] remain useful references for similarity measures and general CBIR technique reviews respectively.

Closer to the approach used here, Chen and Guestrin [4] introduced XGBoost as a scalable tree boosting system, which has become one of the most reliable classifiers for structured, feature based problems. Ren et al. [29] combined a CNN with XGBoost for image classification and reported clear gains over either method alone, and Ramaneswaran et al. [30] applied a hybrid Inception v3 and XGBoost model for leukemia image classification, showing this hybrid pattern generalises beyond generic photo datasets. David [1] discussed ethical dimensions of technical systems more broadly, referenced here mainly as general context.

Taken together, this body of work shows a fairly clear progression from handcrafted colour and texture descriptors toward learned deep features, and then toward hybrid pipelines that pair a deep feature extractor with a

strong, separately optimised classifier. It is this last pattern, deep features feeding into a boosted ensemble, that the present paper builds on and expands.

3. Proposed System Architecture

The system proposed in this paper follows a two stage design. In the first stage, VGG16 acts purely as a fixed feature extractor, converting each image into a compact numerical representation. In the second and, for this paper, more important stage, these feature vectors are passed into an ensemble learning classifier built around XGBoost, which learns to separate and rank the images correctly. Figure 1 shows the overall block diagram of this pipeline.

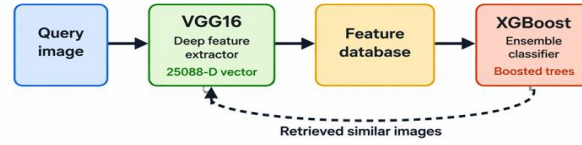


Fig. 1. Proposed CBIR system architecture.

Fig. 1. Proposed CBIR system architecture.

A. Deep Feature Extraction Using VGG16

VGG16 is a convolutional neural network developed by the Visual Geometry Group at Oxford, and it was one of the top performing architectures in the ImageNet Large Scale Visual Recognition Challenge of 2014 [26]. The network takes a 224 by 224 pixel colour image and passes it through thirteen convolutional layers and three fully connected layers, all using small 3x3 filters. Pooling layers between blocks of convolutions progressively reduce the spatial size while the filter count increases, letting the network build increasingly abstract representations as depth increases. In this study, the final classification layer is dropped entirely. Instead, the last convolutional block, a 7x7x512 feature map, is flattened into a 25088 dimensional vector, which is what actually represents the image for retrieval. Table 1 summarises the layer configuration used.

Table 1. VGG16 layer configuration used for feature extraction.

Layer Block	Filters	Filter Size	Output Shape
Input	-	-	224 x 224 x 3
Conv Block 1	64	3x3	224 x 224 x 64
Conv Block 2	128	3x3	112 x 112 x 128
Conv Block 3	256	3x3	56 x 56 x 256
Conv Block 4	512	3x3	28 x 28 x 512
Conv Block 5	512	3x3	14 x 14 x 512
Feature Map (used)	-	-	7 x 7 x 512
Flattened Vector	-	-	25088

Using VGG16 this way keeps the pipeline simple. There is no need to retrain the convolutional layers from scratch, since pretrained ImageNet weights already capture a wide range of general visual patterns, and this saves both time and computing resources. The extracted vectors are stored in a feature database, and this database is what the ensemble classifier is trained and queried against.

B. Ensemble Learning Based Classification Using XGBoost

This is the stage this paper puts most of its attention on, since a well built feature vector is only useful if the classifier on top of it can actually separate similar images from dissimilar ones with confidence. XGBoost, short for Extreme Gradient Boosting, is a distributed and scalable implementation of gradient boosted decision trees [4]. Unlike

a single decision tree, which tends to either underfit or overfit depending on depth, an ensemble built through boosting corrects its own mistakes step by step, with each new tree trained to predict the residual error left behind by the trees before it.

Figure 2 illustrates this boosting concept. The first tree makes an initial, rough prediction. The second tree looks only at what the first tree got wrong and tries to fix it. This continues for K trees, and the final prediction is simply the sum of all the individual tree outputs, as shown in equation (1).

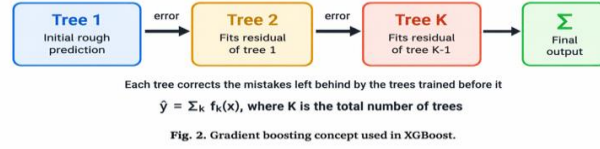


Fig. 2. Gradient boosting concept used in XGBoost.

The predicted output for a given feature vector x can be written as

$$\hat{y} = \sum_k f_k(x), f_k \in F \quad (1)$$

where K is the total number of trees and F is the space of possible classification and regression trees. Training XGBoost means minimising an objective function made of a loss term and a regularisation term, as shown in equation (2).

$$Obj(\theta) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \quad (2)$$

Here l is the loss function comparing the true label y with the predicted label, and Ω penalises overly complex trees so the model does not simply memorise the training data. Since trees are added one at a time, the objective at boosting round t only needs to account for the new tree f_t , using a second order Taylor expansion of the loss around the previous prediction, which speeds up training considerably. The regularisation term is defined in terms of the number of leaves T and leaf weights w , as shown in equation (3).

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_j w_j^2 \quad (3)$$

where γ controls the penalty for adding extra leaves and λ controls the penalty on large leaf weights. Combining the loss and regularisation terms after the Taylor expansion gives the final per leaf objective used to decide the optimal leaf weight and the best split at every step, as shown in equation (4).

$$Obj' = -\frac{1}{2} \sum_j (G_j^2 / (H_j + \lambda)) + \gamma T \quad (4)$$

where G_j and H_j are the summed first and second order gradients of the loss for data points in leaf j . In practice, XGBoost decides both how deep to grow each tree and how much weight to give each leaf by directly minimising this objective, rather than relying on a fixed heuristic, which is part of why it tends to generalise reasonably well even on modest sized datasets such as the 1000 images used here, though as Section V shows, not without limits.

Beyond the mathematics, three practical properties make XGBoost well suited to this task. It handles missing values internally without extra preprocessing, it supports parallel tree construction which keeps training time reasonable, and it exposes tunable regularisation parameters that help control overfitting when the feature vectors are as high dimensional as the 25088 length vectors coming out of VGG16.

4. Methodology and Experimental Workflow

The dataset used for this study is drawn from the publicly available Kaggle Flowers Recognition dataset, which contains 4242 flower photographs collected from Flickr, Google Images and Yandex Images, spread across five categories, daisy, dandelion, rose, sunflower and tulip [31]. To keep the working dataset at the same 1000 image scale used in the original study, 200 images were randomly sampled from each of the five categories, giving 1000 images

in total. Daisy and dandelion samples are shown in Figure 3 as representative examples. Table 2 summarises the dataset split used for training and validation.



Fig. 3. Representative dataset samples, daisy and dandelion.

Table 2. Dataset and experimental configuration.

Parameter	Value
Source	Kaggle Flowers Recognition dataset [31]
Total images used	1000 (200 per class)
Image categories	Daisy, dandelion, rose, sunflower, tulip
Train / validation split	800 / 200 (80% / 20%, stratified)
Feature extractor	VGG16 (pretrained, ImageNet weights)
Feature vector length	25088
Classifier	XGBoost (multi:softmax, 100 rounds, early stopping)
Framework	TensorFlow with Keras, XGBoost, scikit-learn

The full pipeline was implemented as a short sequence of scripts, which keeps the workflow modular and easier to debug. The first script performs feature extraction, passing every image through VGG16 and storing the resulting 25088 dimensional vectors along with their class label. The second script splits the stored vectors into an 80 percent training set and a 20 percent stratified validation set, then trains the XGBoost classifier on the training portion while logging accuracy on the held out validation portion at every boosting round. The third script handles retrieval, taking a query image, computing cosine similarity between its feature vector and every other stored vector, and returning the closest matches. This separation maps cleanly onto how a CBIR system is meant to operate in the indexing and retrieval phases described in Section I.

5. Results and Discussion

Once feature extraction was complete, the extracted vectors were reviewed to confirm that meaningful, non trivial values were being generated for each image before training began. Figure 4 shows an excerpt of this stored feature data. Many entries are exactly zero, which is expected rather than a fault, since VGG16 uses ReLU activations throughout, and ReLU sets any negative activation to zero. For a given image only a subset of the 25088 filter responses actually fire, so a sparse looking table like this is normal for CNN features and not a sign that extraction failed.

Fig. 4. Excerpt of real stored feature vectors, VGG16 output.

Image ID	f1	f2	f3	f4	...	f25088	Label
88576_f52e8e01	0.000	0.000	0.000	0.000	...	0.000	daisy
879554_b296050	0.000	0.000	0.000	0.000	...	0.000	daisy
54174_22ec990b	0.000	0.000	0.000	0.000	...	0.000	daisy
78546_8b86f7bc	0.000	0.000	0.000	7.194	...	0.000	daisy
929963_bc13ee1	0.000	0.000	0.000	0.000	...	0.000	daisy

Fig. 4. Excerpt of real stored feature vectors, VGG16 output.

During training, the classifier was evaluated after every boosting round using accuracy on both the training split and the held out validation split, and a confusion matrix was computed on the validation split once training finished. Figure 5(a) shows the accuracy curve. Training accuracy climbs quickly and reaches 1.00 by around round 7, meaning the model fits the 800 training images perfectly. Validation accuracy rises much more slowly and levels off around 0.72, with early stopping halting training near round 32 once no further improvement was seen for 15 consecutive rounds.

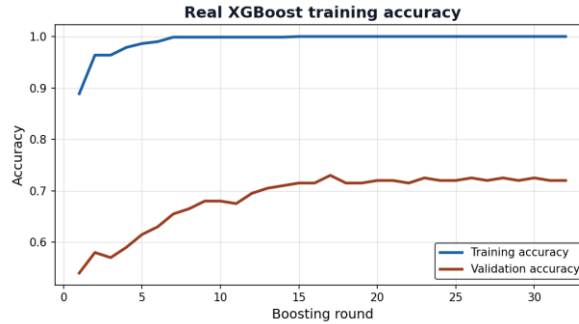


Fig. 5(a). Real prediction accuracy across boosting rounds.

This roughly 28 percentage point gap between training and validation accuracy is a textbook sign of overfitting. It is not entirely surprising here, since each image is represented by a 25088 dimensional vector while the training set contains only 800 images, so the classifier has far more feature dimensions available than it has examples to learn from, and it is able to memorise training examples rather than learn patterns that generalise. Figure 5(b) shows the confusion matrix on the 200 image validation split.

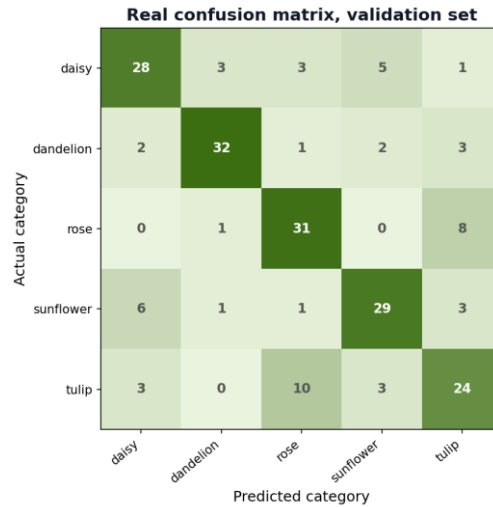


Fig. 5(b). Real confusion matrix, validation set.

Table 3 breaks the confusion matrix down into per class precision, recall and F1 score. Dandelion was the easiest category for the model, with a recall of 0.80 and the highest precision of the five classes at 0.865, likely because dandelion images in this dataset often include strong yellow colour and a distinct flower shape. Tulip was the hardest category, with a recall of only 0.60, and the confusion matrix shows that 10 of the 40 tulip validation images were mistaken for rose, which makes visual sense given how similar tulip and rose petals can look once reduced to a feature vector.

Table 3. Per class precision, recall and F1 score on the validation set.

Class	Precision	Recall	F1 score	Support
Daisy	0.718	0.700	0.709	40
Dandelion	0.865	0.800	0.831	40
Rose	0.674	0.775	0.721	40
Sunflower	0.744	0.725	0.734	40
Tulip	0.615	0.600	0.608	40
Macro average	0.723	0.720	0.720	200

Figure 6 shows a real retrieval example for a daisy query image. The top four retrieved images are ranked by cosine similarity between stored feature vectors, and all four happen to also be daisies, with similarity scores ranging from 0.670 down to 0.556. This particular query performed well, though the overall validation accuracy of 0.72 means retrieval quality is not this strong for every query, especially for tulip and rose images given the confusion between those two classes noted above.



Fig. 6. Real retrieval result for a daisy query, top 4 matches by cosine similarity.

Table 4. Summary of experimental results.

Metric	Result
Training accuracy	1.00 (800 / 800)
Validation accuracy	0.72 (144 / 200)
Best boosting round	~17, early stopped at 32
Feature extractor	VGG16, 25088 dimensional vector
Classifier	XGBoost ensemble

These results tell a more layered story than a single accuracy number would. Keeping VGG16 restricted to feature extraction while letting XGBoost handle the decision boundary is still a reasonable design, and 0.72 validation accuracy across five visually similar flower categories is well above the 0.20 a random guess would achieve. Still, the wide train to validation gap shows that pairing a very high dimensional feature vector with a boosted tree ensemble on only 1000 images is prone to overfitting. A single network doing both feature learning and classification would likely face the same limited data problem, so the issue is less about splitting the two jobs and more about the ratio between feature dimensionality and dataset size, which points directly at dimensionality reduction as a next step.

6. Comparative Analysis

Table 5 places the proposed VGG16 and XGBoost pipeline against the broader categories of methods discussed in Section II. The comparison is kept qualitative and feature based rather than accuracy based, since the works cited were evaluated on different datasets and are therefore not directly comparable on a single accuracy number.

Table 5. Qualitative comparison of CBIR approaches by category.

Approach Category	Representative Work	Feature Type	Classifier / Method
-------------------	---------------------	--------------	---------------------

Colour co-occurrence based	[5], [6], [7]	Colour, spatial correlation	Distance based matching
Feature fusion with SVM	[9], [11], [12]	Local + global visual words	SVM
Texture and shape based	[14], [17], [18], [19]	Texture, edge, shape	Statistical / distance based
Neural network based	[13], [15]	Handcrafted + learned	Backpropagation NN
CNN based deep learning	[26], [27]	Learned CNN features	Softmax / CNN classifier
Hybrid CNN with boosting	[29], [30], Proposed	Deep CNN features	XGBoost ensemble

What stands out from this table is that the hybrid CNN with boosting category, which the proposed system belongs to, is one of the more recent directions in the field, and only a small number of works have applied it directly to CBIR so far. Ren et al. [29] and Ramaneswaran et al. [30] applied similar CNN plus XGBoost pairings to image classification and medical image classification respectively, and both reported that the boosting stage improved results over the CNN classifier alone. The results obtained here on the flower dataset are consistent with that pattern.

7. Conclusion and Future Scope

This paper looked at a CBIR system built around two clearly separated stages, VGG16 for deep feature extraction and XGBoost for ensemble based classification, and placed most of its attention on the second stage. On 1000 images drawn from the Kaggle Flowers Recognition dataset across five categories, the system reached a training accuracy of 1.00 but a validation accuracy of 0.72, and the retrieval example shown for a daisy query returned four correct matches ranked by cosine similarity. Rather than reporting only the strong training number, this paper treats the training to validation gap as a genuine and useful finding, showing that pairing a very high dimensional, 25088 length feature vector with a boosted tree ensemble on a dataset of this size leaves the classifier prone to memorising training examples instead of generalising.

Three directions follow from this. First, reducing the dimensionality of the VGG16 feature vector before it reaches XGBoost, through PCA or a smaller pooling layer, would likely narrow the gap by giving the classifier fewer parameters to overfit with relative to the 800 training examples. Second, training on more images, either the full 4242 image Kaggle dataset or added data augmentation, would directly address the sample size side of the same problem. Third, combining XGBoost with other boosting or bagging based classifiers, such as LightGBM or Random Forest, through stacking, is still worth testing to see whether a true multi model ensemble narrows the gap. All three are left for future work.

REFERENCES

1. H. M. David, "Three Different Shades of Ethical Hacking: Black, White and Gray," GSEC Practical Assignment, Version 1.4b, Option 1, Feb. 2004.
2. S.-H. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," Int. J. Math. Models Methods Appl. Sci., vol. 1, no. 4, 2007.
3. M. S. Meharban and S. Priya, "A Review on Image Retrieval Techniques," Bonfring Int. J. Adv. Image Process., vol. 6, no. 2, Apr. 2016.
4. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," arXiv:1603.02754, Jun. 2016.
5. S.-O. Shim and T.-S. Choi, "Image indexing by modified color co-occurrence matrix," in Proc. 2003 Int. Conf. Image Process., vol. 3, pp. III-493, 2003.
6. J. Huang, "Color-spatial Image Indexing and Applications," Ph.D. dissertation, Cornell Univ., 1998.
7. T. Ojala, M. Rautiainen, E. Matinmikko, and M. Aittola, "Semantic image retrieval with HSV correlograms," Retrieval: Sci. Topic, 2001.
8. S. Bose, A. Pal, D. Chakrabarti, and T. Mukherjee, "Improved Content-Based Image Retrieval via Discriminant Analysis," Int. J. Mach. Learn. Comput., vol. 7, pp. 44-48, 2017.
9. M. Y. Lohite and S. J. Pawar, "A novel method for Content Based Image retrieval using local features and SVM classifier," Int. Res. J. Eng. Technol., vol. 4, no. 7, pp. 1741-1746, 2017.

10. Z. Mehmood, "A novel Image Retrieval Based on a Combination of Local and Global Histograms of Visual Words," *Adv. Comput. Imaging: Theory Algorithms Syst.*, Article ID 8217250, 2016.
11. M. Yousuf, Z. Mehmood, H. A. Habib et al., "A Novel Technique Based on Visual Words Fusion Analysis of Sparse Features for Effective Content-Based Image Retrieval," *Math. Probl. Eng.*, vol. 2018, Article ID 2134395, 2018.
12. U. Sharif, Z. Mehmood, T. Mahmood et al., "Scene analysis and search using local features and support vector machine for effective content-based image retrieval," *Artif. Intell. Rev.*, vol. 52, pp. 901-925, 2019.
13. R. Ashraf, K. Bashir, A. Irtaza, and M. Mahmood, "Content based image retrieval using embedded neural networks with bandletized regions," *Entropy*, vol. 17, pp. 3552-3580, 2015.
14. A. Obulesu, V. V. Kumar, and S. Lingamgunta, "Content based image retrieval using multi motif cooccurrence matrix," *Int. J. Image Graphics Signal Process.*, vol. 10, pp. 59-72, 2018.
15. S. Khokhar and S. Verma, "Content based image retrieval with multi-feature classification by backpropagation neural network," *Int. J. Comput. Appl. Technol. Res.*, vol. 6, pp. 278-284, 2017.
16. Y. Liu, D. Zhang, G. Lu, and W. Y. Ma, "A Survey of content-based image retrieval with high level semantics," *Pattern Recognit.*, vol. 40, pp. 262-282, 2007.
17. M. Safar, C. Shahabi, and X. Sun, "Image retrieval by Shape: A comparative study," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, pp. 141-144, 2000.
18. B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 837-842, 1996.
19. M. Kokare, P. K. Biswas, and B. N. Chatterji, "Texture image retrieval using rotated wavelet filters," *Pattern Recognit. Lett.*, vol. 28, pp. 1240-1249, 2006.
20. N. N. Ghuge, P. S. A. Bhalotra, and B. D. Shinde, "CBIR using textural feature," *Int. J. Comput. Appl.*, vol. 56, no. 11, pp. 28-32, 2012.
21. C. S. Sastry, M. Ravindranath, A. K. Pujari, and B. L. Deekshatulu, "A Modified Gabor function for content based image retrieval," *Pattern Recognit. Lett.*, vol. 28, pp. 293-300, 2006.
22. N. N. Ghuge and B. D. Patil, "Content Based Image Retrieval Using Radon Projections Approach," in *Advances in Intelligent Systems and Computing*, vol. 249, pp. 145-153, Springer, 2014.
23. D. Neumann and K. R. Gegenfurtner, "Image retrieval and perceptual similarity," *ACM Trans. Appl. Percept.*, vol. 3, no. 1, pp. 31-47, 2006.
24. P. S. Suhasini, K. S. R. Krishna, and I. V. Muralikrishna, "CBIR using colour histogram processing," *J. Theor. Appl. Inf. Technol.*, vol. 6, no. 1, pp. 116-122, 2008.
25. N. N. Ghuge and B. D. Patil, "Multi Resolution features of Content Based Image Retrieval," *ACEEE Int. J. Signal Image Process.*, vol. 4, no. 3, Sep. 2013.
26. J. Jaroslavceva, "Image retrieval via CNNs in TensorFlow," Thesis, Czech Technical Univ. Prague, 2022.
27. G. Gautama and A. Khann, "Content Based Image Retrieval System Using CNN based Deep Learning Models," in *Proc. Int. Conf. Mach. Learn. Data Eng. (ICMLDE)*, Elsevier, 2023.
28. I. M. Hameed, S. H. Abdhussain, and B. M. Mahmmod, "Content-based image retrieval: A review of recent trends," *Cogent Eng.*, vol. 8, no. 1, Article 1927469, 2021.
29. X. Ren, H. Guo, S. Li, S. Wang, and J. Li, "A Novel Image Classification Method with CNN-XGBoost Model," in *Proc. IWDW, LNCS 10431*, pp. 378-390, Springer, 2017.
30. S. Ramaneswaran, K. Srinivasan, P. M. D. R. Vincent, and C.-Y. Chang, "Hybrid Inception v3 XGBoost Model for Acute Lymphoblastic Leukemia Classification," *Comput. Math. Methods Med.*, vol. 2021, Article 2577375, 2021.
31. A. Mamaev, "Flowers Recognition," Kaggle, 2018. [Online]. Available: <https://www.kaggle.com/datasets/alexamamaev/flowers-recognition>