Intelligent Systems with Applications 16 (2022) 200108

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Fusion of vectored text descriptors with auto extracted deep CNN features for improved image classification

Sudeep D. Thepade *, Jovian A. Jaison

Computer Engineering Department, Pimpri Chinchwad College of Engineering, SPPU, Pune, India

ARTICLE INFO

Keywords:
Content-based image classification
Pre-trained dcnn
LSTM
Text embedding

ABSTRACT

In today's age, massive image data is generated rapidly. This influx has made labeling images tedious and, in turn, made it harder to retrieve images through searching algorithms that rely only on labels, keywords, or other meta-data in the images. Modern Content-Based Image Retrieval (CBIR) techniques rely on the visual features within the image to return relevant results to a search query. Deep Convolutional Neural Network (DCNN) models made great strides in the last decade. This paper relies on these complex pre-trained models to extract visual features from images. The proposed work has used pre-trained models like VGG16, MobileNet, Inceptionv3, and Xception for this task. Some studies in the CBIR space also suggest increased accuracy when both visual and textual features are considered. This paper proposes a novel three-step process for obtaining textual features. Firstly, the proposed model receives keywords for each image using Google Cloud Vision API. Secondly, the proposed model replaces each keyword with a 300-dimensional embedding vector obtained using word2vec, trained on the Google News dataset. Finally, the proposed model trains a combination of the Deep Semantic Similarity Model (DSSM) and Long Short-Term Memory (LSTM) model to reduce the 300-dimensional vector to a 64-dimensional vector. Using these new shortened word vectors, the proposed model computes the cosine similarity to replace each keyword of an image with five of its synonyms. Here, these additional steps increased the accuracy compared to simply using a word embedding technique. Finally, the proposed model combined the visual and textual feature vectors and observed that this feature set showed maximum classification accuracy of 98.33%, which is also compared with and found relatively better than other similar model results.

Introduction

Classification is the problem of predicting which category an observation belongs to among a set of categories. A classification problem may be a supervised problem, in which case, while training a statistical model, both the list of observations and the categories to which they belong are supplied as inputs to train the model. Or it may be an unsupervised problem where only the list of observations is provided as an input, and methods like clustering are used to find patterns in the data and group them into categories. In today's age, a large amount of image data is being generated rapidly due to factors like the advent of social media and increased accessibility to smartphones having cameras. One of the significant problems that this flooding of image data has given rise to is that traditional image retrieval methods that used labels, file names, or other metadata to return search results have become cumbersome and inefficient. Even on local devices like phones assigning file names to each image manually to signify what it contains is tedious; also, assigning these labels is an ambiguous process as it would be solely based on an individual's vocabulary preference and biases. Content-Based Image Classification (CBIC) is one of the solutions to this problem. It uses the content within the images like color or texture features, objects identified within the image, etc., to make classification and retrieval much more efficient by today's needs and standards.

Text descriptors are used to convey what an image contains. They are manually added keywords or phrases used to describe an image or supplementary information associated with the image, like captions or subtitles. These text descriptors can be valuable features for CBIC; however, they cannot be used in their textual form directly for classification, so the text descriptors are vectorized. Word vectorization is a technique for mapping phrases or keywords to a corresponding numerical vector which can then be used along with distance measures like cosine similarity or euclidean distance for finding word similarities. A combination of the Deep Semantic Similarity Model (DSSM) and Long Short-Term Memory (LSTM) is increasingly being used for text-based

* Corresponding author.
E-mail address: sudeepthepade@gmail.com (S.D. Thepade).

https://doi.org/10.1016/j.iswa.2022.200108
Received 23 October 2021; Received in revised form 23 July 2022; Accepted 6 August 2022
Available online 11 August 2022

image classification (Mahalakshmi et al., 2020). DSSM is a word vectorization technique that uses Deep Neural Net (DNN) modeling to represent labels, keywords, or phrases in a continuous semantic space and model semantic similarity between the strings. LSTM networks are a special type of Recurrent Neural Network (RNN) crafted to overcome the long-term dependency problem. The long-term dependency problem is observed in practice in RNNs where they cannot retain important information if the gap between relevant information and the point where it is needed is substantial (Figs. 1–8).

A pre-trained model is a deep learning architecture created and trained beforehand. Typically these pre-trained models are trained on large datasets and have complex state-of-the-art architectures that make them very accurate but difficult to train on a typical consumer-oriented low-powered system. When a neural network is trained, the refined weights get assigned to each connection between neurons in a network to obtain maximum classifying accuracy. Instead of retraining a pre-trained model on the dataset for our problem statement, one can leverage what the complex network has already learned from training on a similar larger dataset. One can directly use the weights of the pre-trained model and its architecture to make predictions for our specific problem statement. This approach is called transfer learning (Thepade et al., 2022). Additional methods include unfreezing particular higher layers of the pre-trained model or replacing the softmax layer at the end of the model with a trainable dense layer to fine-tune the pre-trained model. One can also use a pre-trained model as a feature extractor (Subhadip Maji et al., 2021). Two significant limitations of using pre-trained models and transfer learning are negative transfer and overfitting. If the initial and target problem statements are not similar enough, the model might perform worse than expected. The accuracy of the new fine-tuned model is lesser than the original model; this is called negative transfer. When the model fits too closely to the training data and memorizes noise, it is said to be 'overfitted.' One of the reasons this may occur is the model is too complex. In an attempt to fine-tune a pre-trained model, if too many dense layers with too many neurons are added, it may lead to overfitting.

Many recent approaches in CBIC (Messina, Amato & Carrara, 2020) use a fusion of visual features like color, texture, and textual features in embedding vectors. This fusion gives better accuracy and provides the increasing capability of the model to understand relations among features, increasing accuracy.

The key contributions of the work presented here are:

- Using transfer learning to fine-tune pre-trained DCNN models for the Wang dataset. These models are then used as auto visual feature extractors by removing their final softmax layer.
- Using word2vec to create a numerical embedding vector for each keyword associated with an image, followed by a combination of DSSM and LSTM model to reduce the size of this word vector. To improve classification accuracy, five synonyms for each keyword are computed by comparing the distance measure of vector pairs.
- Inclusion of visual and textual features in the classification process by fusing auto-extracted features from pre-trained models and word to vector features obtained from word2vec, DSSM, and LSTM for improved CBIC.

The rest of the paper is organized as follows. Section 2 consists of the

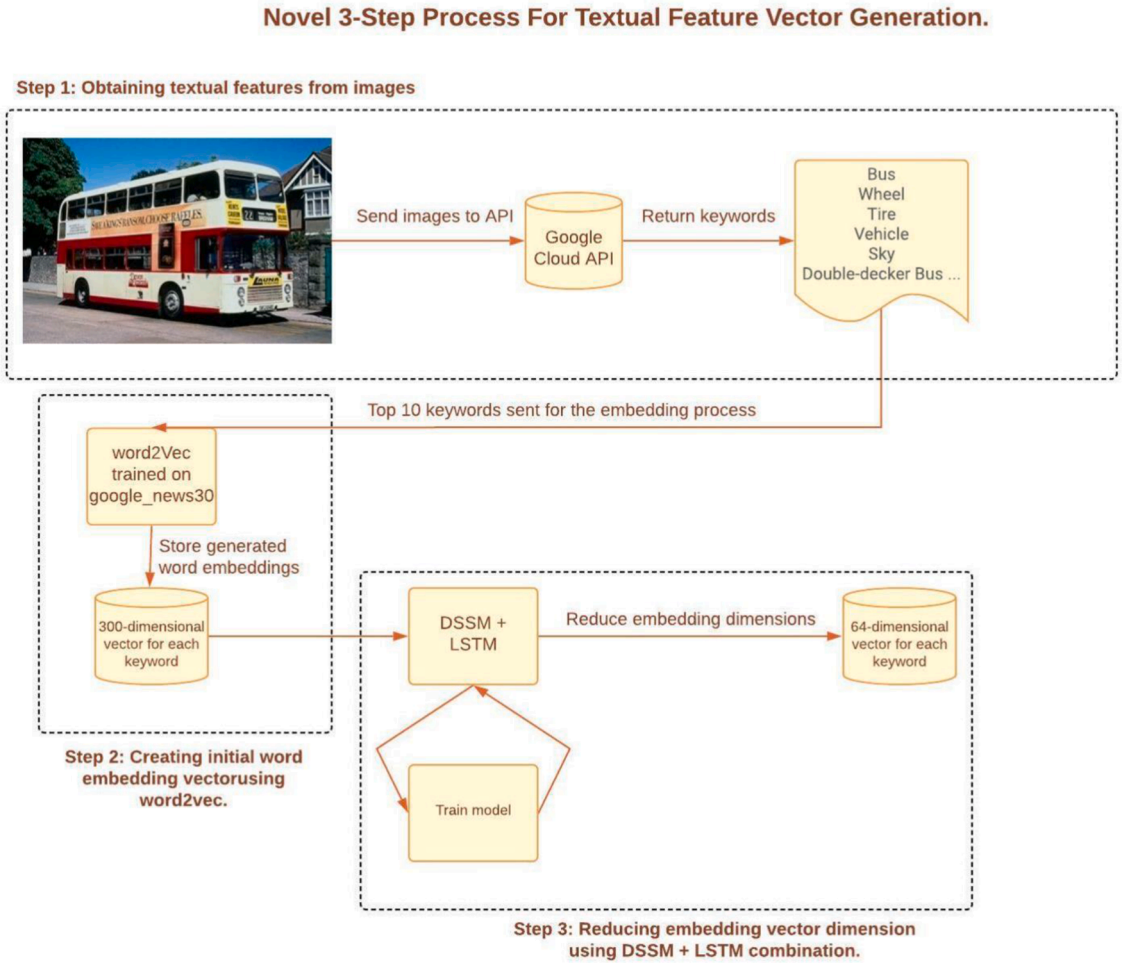

Fig. 1. Illustration of the novel 3-step process for creating textual features (Color image).

```
[ ]  v1 = word2vec.wv['people']
     print(v1)

     [-1.5806822e-03 -4.7377823e-03 -4.4460562e-03 -2.8618334e-03
       4.5191376e-03 -2.0074227e-03 -1.9792712e-03  4.7569387e-03
       3.2272181e-03  3.8931037e-03  1.2926498e-03 -2.2483780e-04
       2.9458798e-04 -3.3519443e-03  2.1080226e-03 -1.2972133e-03
       2.0405431e-03 -4.4068671e-03  1.1388600e-03  5.9078663e-04
      -3.5805986e-04 -1.6196191e-03  3.3567245e-03  3.4922757e-03
       1.8710227e-04 -9.4402122e-04 -4.5420146e-03 -2.6799550e-03
       7.8887460e-05  4.5544608e-03 -2.2203177e-03  2.7128716e-04
      -1.6097175e-03  3.7126297e-03  4.3614549e-03 -2.0190182e-03
      -1.8596520e-03  3.0201094e-03 -3.5294979e-03  2.9625315e-03
      -2.3555593e-03 -9.4132905e-04 -4.7619762e-03  4.3783830e-03
       1.5222902e-03  3.6754115e-03  4.1129957e-03 -2.0116286e-03
       2.0040452e-04  3.7493235e-03 -4.6881409e-03 -2.8597314e-03
      -6.2440633e-04  4.5048585e-03 -6.6684722e-04 -2.7103964e-03
       3.4140775e-03  4.2267591e-03  2.7126384e-03 -3.3497082e-03
      -2.6758066e-03  1.0630045e-03  4.3916758e-03  4.6758987e-03
       1.3373037e-03  3.7236523e-03  1.4883729e-05 -2.2907839e-03
       1.1776722e-03  2.2815466e-03  2.7677843e-03 -2.2887860e-03
      -1.5969191e-03  2.6198537e-03  1.2752647e-03  1.7110729e-03
       4.5571178e-03  4.5158127e-03  5.5595749e-04  1.9995417e-03
      -9.7091630e-04  2.5690964e-03 -3.2056081e-03 -2.8098577e-03
      -9.0523441e-05 -8.6983974e-04 -4.4037453e-03 -1.2917650e-03
       2.6633192e-03  2.6481876e-03  3.9412184e-03  4.8398990e-03
      -1.1343103e-03  3.3933218e-03 -2.7411201e-03  1.0632530e-03
       4.3835202e-03  2.9934519e-03 -7.9378422e-04  1.6746494e-03]
```

Fig. 2. Illustration of a 300-dimensional vector obtained for the word 'people' from word2vec trained on Google News. (Color image).

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1)]               0
_________________________________________________________________
embedding (Embedding)        (None, 1, 300)            239700
_________________________________________________________________
bidirectional (Bidirectional (None, 1, 128)            186880
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                41216
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650
=================================================================
Total params: 472,606
Trainable params: 472,606
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3. Architecture of the proposed DSSM-LSTM model initialized with the Google News weights. Output 64-dimensional vectors are taken from the 'dense' layer. (Color image).

literature study conducted for the work presented here. The experimentation environment set up for validating the proposed method is stated in Section 3. The proposed method for the paper is discussed in Section 4. Section 5 puts forth the results obtained from experimentation. Finally, the conclusion is provided for the explored and presented work.

Literature survey

Many research papers have proposed interesting approaches for CBIC using either DSSM with LSTM, pre-trained deep learning models, or a fusion of text and content-based features. In this section, a brief review of these works is provided.

The model, which performs two different operations i) Retrieval of Text ii) Retrieval of Images, is proposed in Mahalakshmi et al. (2020). For the former, they used a bi-directional LSTM, which was then optimized using a differential evolution algorithm. For the latter, they used a CNN whose architecture was based on the Residual Net 50 model. Along with this model, they used Manhattan distance as the similarity measure. The work by Messina et al. (2020) proposes a relational

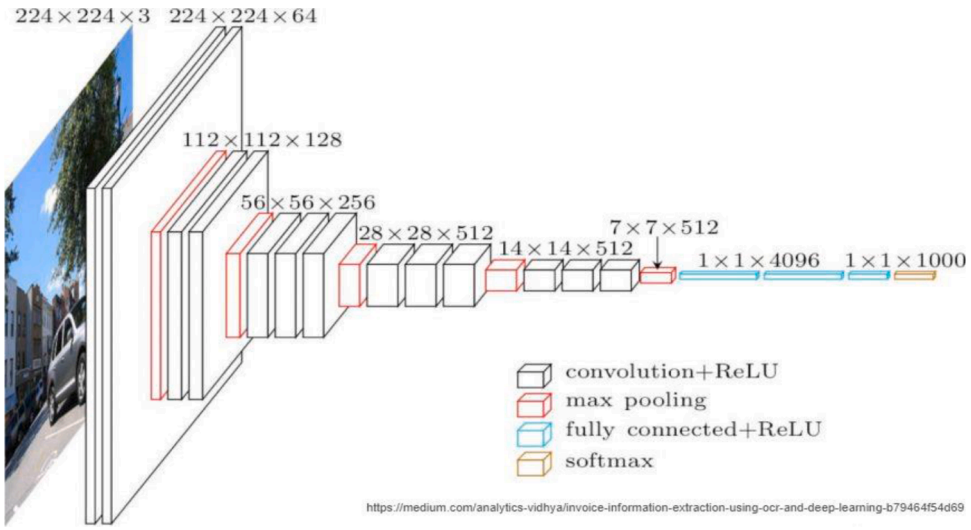

Fig. 4. Illustration of the VGG16 model architecture. Image from (VGG16 Image Reference).

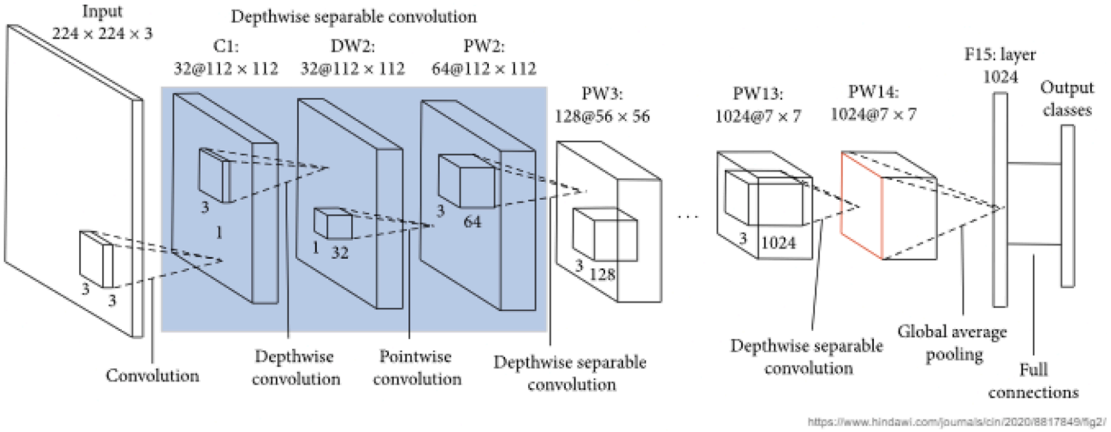

Fig. 5. Illustration of the MobileNet model architecture. Image from (MobileNet Image reference).

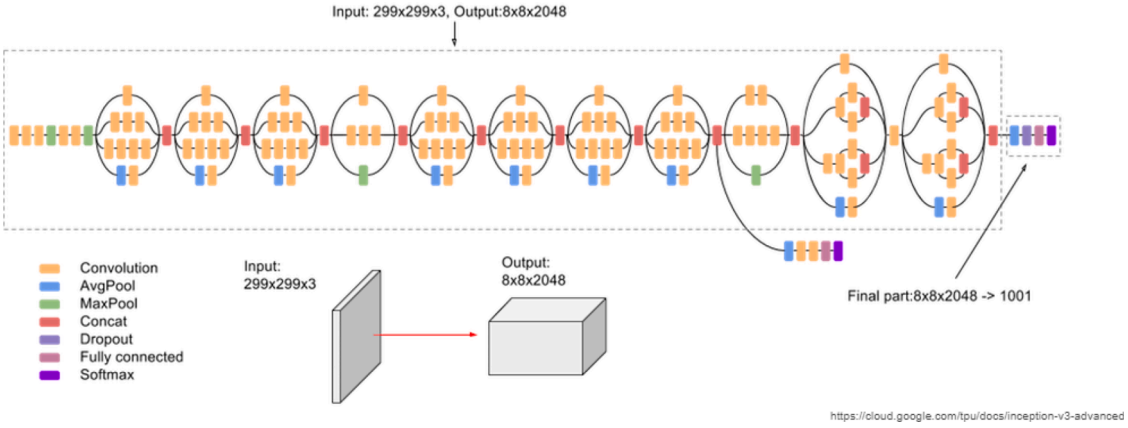

Fig. 6. Illustration of the Inceptionv3 model architecture (Inceptionv3 image reference).

content-based image retrieval system to return images with similar inter-object relationships. They present a novel Aggregated Visual Features Relation Network (AVF-RN) that performed better than the existing Two-Stage Relation Network (2S-RN) module. Both the models were trained on Relational Visual Question Answering (R-VQA). Still, AVF-RN performed better as it produced better relation-aware features by learning aggregation inside the network itself. Both 2S-RN and AVF-RN used LSTM to generate embedding vectors for each question and paired it with a CNN for effectively learning inter-object relationships through R-VQA. In Hamid et al. (2016), the authors propose sentence embedding using RNN with LSTM cells. The LSTM-RNN combination converts each word in the sentence into a semantic vector sequentially. The LSTM

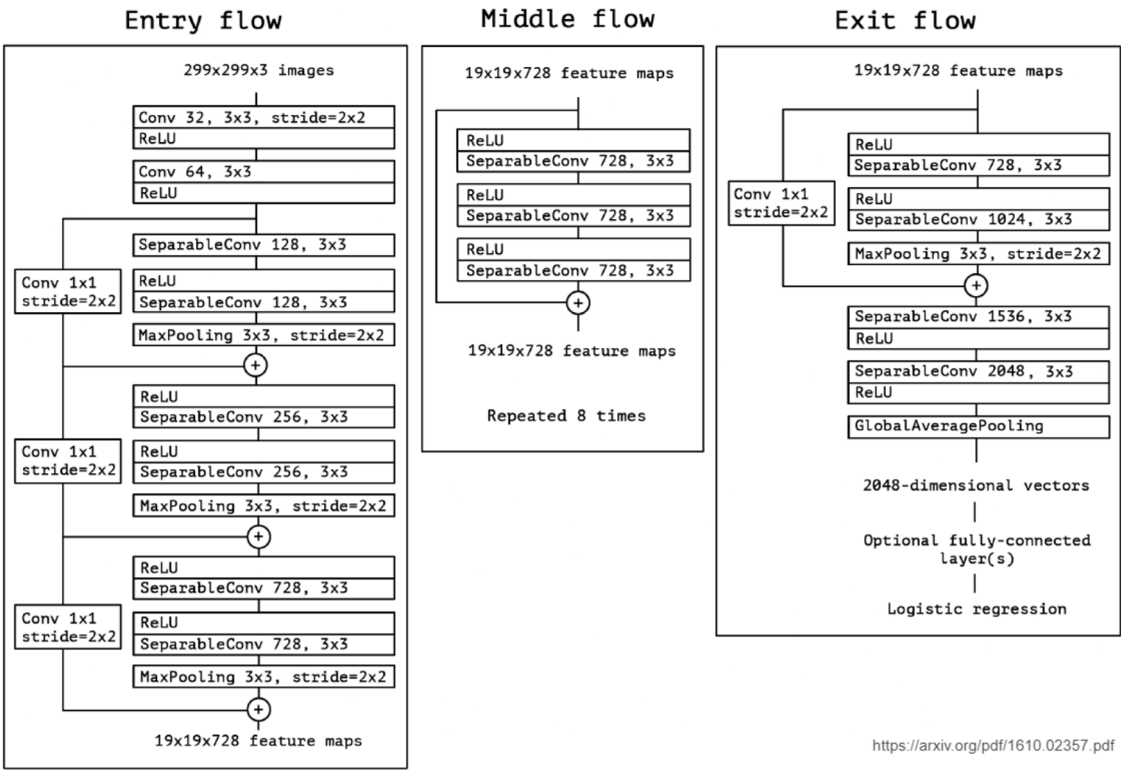

Fig. 7. Illustration of the Xception model architecture Image (Xception Image reference).

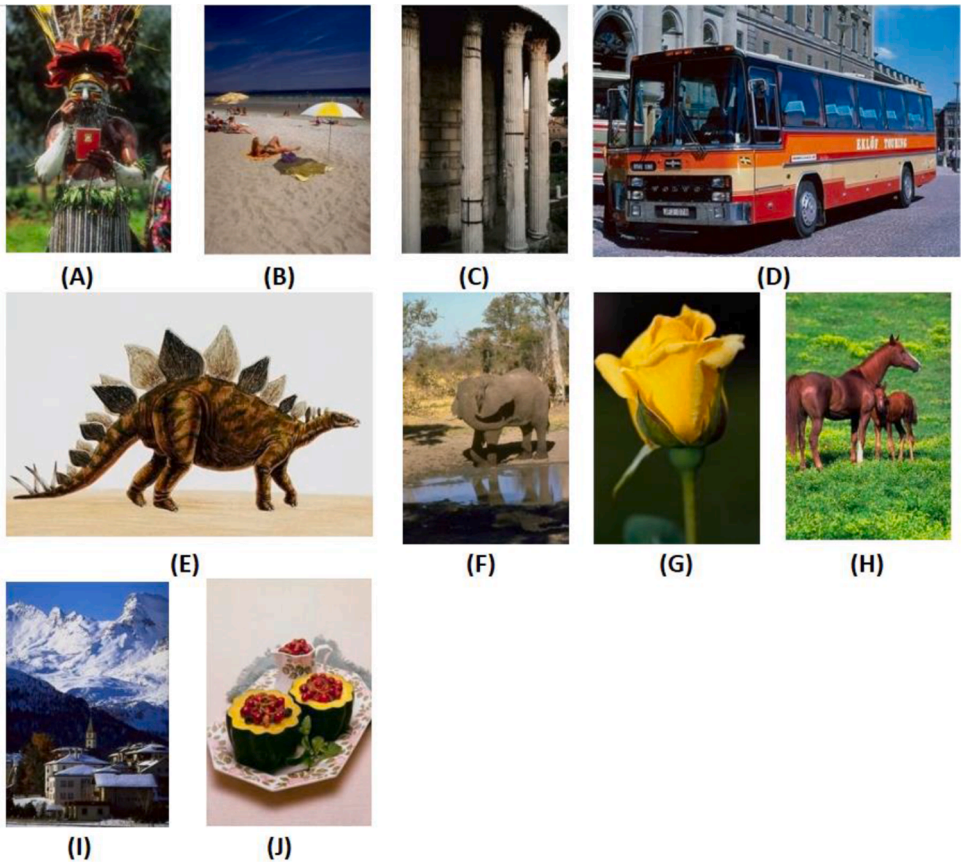

Fig. 8. Image samples from the Wang dataset (James Z. Wang et al., 2001) (Jia Li et al., 2003); [A] Tribes [B] Beach [C] Architecture [D] Bus [E] Dinosaur [F] Elephant [G] Flower [H] Horse [I] Mountain [J] Food.

helps to retain the information of the entire sentence and provides its corresponding semantic representation. The proposed system is helpful in web document retrieval. A comparison with another popular sentence embedding method called Paragraph vector shows that the proposed method performed better at the document retrieval task.

The method proposed by Subhadip Maji et al. (2021) makes use of features obtained from pre-trained deep convolutional networks that were trained on large image classification datasets for CBIR. Xception, Inception v3, NasNet Large are some of the pre-trained models used in their study. They removed the last softmax activation layer from each model to make the final classification based on features extracted by the neural network. Obtaining the feature vector from this layer gave them the most learned high-level features as each model's deepest layer. Furthermore, they observed the retrieval times of the system and proposed a pre-clustering method for the image database to reduce retrieval time. In the paper (Bhandi et al., 2019), the authors use the VGG16 and ResNet pre-trained models for CBIR. They obtain feature vectors from the second last dense layers of both models, i.e., 4096 VGG16 features and 2048 ResNet features, and club them. This fusion of vectors from pre-trained DCNN models and an offline database indexing module and online query module are the novelties of this study. In Barbhuiya, Karsh and Jain (2021), the authors make use of CNN for Hand Gesture Recognition (HGR), where static signs consisting of both alphabets and numerals of the American Sign Language (ASL) are considered. Modified versions of pre-trained VGG16 and AlexNet are used for feature extraction. These features are then supplied as input to an SVM for classification. Different layers of the pre-trained models were considered for feature extraction, and their accuracies were compared for best performance.

A content-based image retrieval system proposed in (Unar, Wang, Wang & Wang, 2019) combines visual and text-based features for retrieval. The system classifies a query image into two categories based on whether or not the text appears within the image. In images where text is present, it is detected and added to a bag of words. If no text is detected, only visual features from the image are extracted. Then a fusion of textual and visual features is performed, and similar images are returned based on the similarity to the fused vector. The system supports three types of queries: image, keyword, and combination. In work proposed by (Huang, Wang, Li & Ning, 2019), the authors study the effect of using textual and visual features on classification. They used a CNN architecture trained to classify whether or not the image in a dataset of tweets was flood-related. They then performed a sensitivity test to obtain flood-sensitive keywords from the tweet. These keywords were used to refine the CNN classification results. It was observed that combining the CNN classification results with the flood-sensitive keyword in the tweets led to a notable increase in precision.

The authors (Li et al., 2019) present a new zero-shot event detection method using semantic search, concept selection, and a new Event-Adaptive Concept Integration (EACI) algorithm. The paper made use of the TRECVID dataset. First, a query event description is provided in the semantic search, i.e., a short sentence or an event name. Then, using word2vec, numerical vectors are obtained for the event description and concepts in the system vocabulary. The concepts in the system vocabulary are then ranked based on the measure of distance between its vector and the query description vector. WordNet is also used in combination with word2vec to improve event description. Finally, EACI calculates the efficacy of semantically related concepts by assigning different weights utilizing the area under the score curve to compute the weights.

The authors (Li et al., 2018) propose a new technique to improve drawbacks in traditional Spectral Clustering-based methods for computer vision tasks. With the new approach, a rich affinity matrix with numerous features can be learned to find the optimum weights for each feature. The affinity weights are given by data. Also, a series of optimum projection matrices are assigned for each feature, which decides the lower dimension space and optimum affinity weight for every pair of data in that space. The affinity weights and lower dimension space are jointly optimized. In work given by (Li et al., 2018), a rank constrained Spectral Clustering method is proposed with a versatile embedding framework. An adaptive probabilistic neighborhood learning process obtains an ideal graph's block diagonal affinity matrix. The flexible embedding obtains the cluster structures in lower-dimensional space and suppresses the noise and irrelevant data in higher-dimensional areas. The block diagonal affinity matrix is learned along with the construction of the adaptive graph. Because of the rank constraints, the count of clusters is bound to converge at ground truth.

In the method proposed by (Mahalakshmi et al., 2020), the Yahoo corpus dataset is used to test the proposed DE-BiLSTM method. However, labels specific to the images in the dataset are not used to evaluate this technique. These nonrelevant labels make the system lesser reliable. Also, in the method proposed by (Mahalakshmi et al., 2020), instead of LSTM, Bidirectional LSTM was applied. BiLSTM comprises of two LSTMs, the first taking the input in a forward direction and the other taking it in a backward direction. Because of this increased information, it provides more context and increases learning.

In the work presented by (Messina et al., 2020), on the other hand, the CLEVR dataset is used. The dataset consists of 3D rendered scenes and corresponding question answers in functional programs but can obtain textual features and relationships relevant to the image or scene. In work done by Messina et al. (2020), the generation of functional programs can work in cases where the image is simple, like the 3D object images used in the paper. However, the process may take longer and perform variably for more complex images and real-life scenarios where numerous objects are present.

The approach given in Hamid et al. (2016) proposed an innovative solution for the sentence embedding problem and proposed using LSTM-RNN models, which could accumulate richer information even in long sentences while attenuating unimportant words. This solution resulted in embedding vectors that were much more accurate and outperformed other methods at the time on tasks like web document retrieval. Here the textual features involve large sentences making it noncompatible for fusion with visual features.

The existing studies are generating nonrelevant text descriptions of images in the form of sentences. These sentence forms of textual description features make it challenging to get converted into a numerical form for possible fusion with visual content features. Most of the existing models are experimented with datasets having images with a single object and simple background, for more generic images with the presence of multiple objects and complex backgrounds; explorations are needed.

Proposed method

This section explores the principal components of the proposed CBIC model. There are four main components: obtaining textual features from the images using the Cloud Vision API, creating embedding vectors from the textual features using word2vec and DSSM-LSTM combination for increased accuracy, obtaining visual features of the images from fine-tuned and pre-trained DCNN models, and finally combining the visual feature vectors with word embedding vectors.

Obtaining textual features from images

Google Cloud offers a Vision API (Google Cloud Vision API online link), which provides powerful pre-trained machine learning models. It can be accessed using REST and RPC API calls. The API offers numerous services like classifying images into predefined categories, detecting objects and faces in an image, recognizing handwritten text, and many more. However, proposed model used this API to assign labels to images for this paper. The API removes any ambiguity or biases while manually labeling the images or assigning textual metadata. The pre-trained models that the API employs help obtain labels and keywords relevant

and specific to each image in the dataset. Each image in the Wang dataset was assigned a maximum of 10 keywords using this technique.

Creating word embedding vectors

Keywords or labels cannot be used directly in their textual form for classification. Instead, they need to be converted to their corresponding numerical vectors using word vectorization techniques. This paper used the pre-trained word2vec module from the Genism library for Python. The Google News dataset consisted of about 100 billion words and 300 million words and phrases to train this pre-trained model. A 300-dimensional vector is obtained for each word or phrase supplied as an input to the model. Next, the labels for each image obtained in the previous step were replaced with their corresponding 300-dimensional vector representation.

Additionally, the proposed model attempted to improve the accuracy of the word2vec embedding by adding a module of DSSM LSTM after it. The proposed model initialized the DSSM layer with the word2vec embeddings and then trained the entire model so that the auxiliary learning would make the vectors more accurate, and the 300-dimensional vector could be condensed into a 64-dimensional vector. Finally, the proposed model uses cosine similarity with these new embeddings to find synonyms for each keyword assigned to an image. The proposed model then replaced each label from the previous step with five synonyms. This 3 step process of word2vec followed by DSSM-LSTM and synonym addition for obtaining text embeddings for CBIC is our novel contribution as compared to other papers like Hamid et al., (2016), Mahalakshmi et al. (2020), and Messina et al. (2020) in a similar space.

Extracting visual features from the images

The method proposed by Subhadip Maji et al. (2021) describes how pre-trained deep learning models can obtain images' features by simply removing the final softmax activation layer. They made use of pre-trained models like DenseNet, InceptionResNetv2, Inception v3, MobileNetv2, NasNet Large, ResNet50, VGG19, Xception. In Bhandi et al., (2019), VGG and ResNet50 were utilized for feature extraction. In work done by Barbhuiya et al. (2021), AlexNet is utilized for feature extraction; these features were then sent to an SVM model for the classification task. This paper used Xception, VGG16, MobileNet, and Inception v3 for feature extraction.

VGG16 (Simonyan, Karen et al., 2014) is a CNN model that achieved a 92.7% top-5 accuracy on the ImageNet dataset, consisting of 14 million images and 1000 classes. It improved over AlexNet by replacing the large kernel-sized filters in the first and second convolutional layers with many 3×3 filters. The input is a 224×224 RGB image.

MobileNet (Andrew et al., 2017) carries out just one convolution for each color channel instead of combining the three and flattening them, i.e., it uses depth-wise separable convolutions. In standard convolution, both filtration and the combination of inputs into a new output are done in one step. In MobileNet, depth-wise separable convolutions split these steps into filtering and combination layers. This step drastically reduced computation and model size.

Inceptionv3 (Christian, Vincent, Sergey, Jon & Wojna, 2016) is the 3rd version of a Deep convolutional architecture by Google and is trained on the 1000 classes of ImageNet dataset on which it obtained greater than 78.1% accuracy. It is 42 layers deep, and the computation cost is much more efficient than VGGNet.

Xception (Chollet, Francois., 2017) stands for an extreme version of Inception. It features a modified version of the depth-wise convolution, better than Inceptionv3. It has a point-wise convolution followed by depth-wise, i.e., It performs 1×1 convolution first and then a channel-wise spatial convolution. Also, the modified depth-wise separable convolution does not have any intermediate ReLU non-linearity.

Combining visual features and text embeddings

The proposed model perform two fusion operations after obtaining visual feature vectors from the pre-trained DCNN models and textual feature vectors (word embeddings) from the novel 3-step process. The proposed model first fuses the visual feature vectors with the 300-dimensional word embeddings and, secondly, the visual feature vectors with the reduced 64-dimensional embeddings. The proposed model then uses multiple classification algorithms of different families, namely, Random Forest, SVM, Decision Trees, and Naive Bayes, to calculate the effectiveness of the features in classifying the images. Finally, the proposed model compares the effectiveness of these two final concatenated vectors and also considers only visual feature vectors and only 300-dimensional and 64-dimensional word embeddings. This way, which feature set gives the best accuracy can be observed.

Comparison with previous studies

In the method proposed by Mahalakshmi et al. (2020), the Yahoo corpus dataset is used to test the proposed DE-BiLSTM method. However, labels specific to the images in the dataset are not used to evaluate this technique. The proposed method addresses this problem by generating the description keywords which are relevant and specific to the image using the Cloud Vision API, ensuring maximum classification accuracy as images of the same classes have the same or closely related keywords.

The CLEVR dataset used by (Messina et al., 2020) having 3D rendered scenes and corresponding question answers in functional programs is obtaining the textual features relevant to the image or scene. But the generation of functional programs can only work in cases where the image is simple with uniform background and only one object. The proposed method may deal better with the more generic and complex images and real-life scenarios where numerous objects are present.

The models given by Mahalakshmi et al. (2020) and Messina et al. (2020) do use variations of LSTM for obtaining the textual description features of images in form of sentences, making it more complex to be fused with the visual content features. The proposed method makes the textual description easier and more relevant to image contents. The proposed method only uses single words, which does not need to use LSTM-RNN combination or a Bi-LSTM system.

The proposed method needed texts to be converted to numerical vectors, which is done with word2vec (Google word2vec online link). However, the vectors obtained were 300-dimensional. This length meant that any processing would be very time-consuming as each image had ten keywords, and each keyword replaced with a 300-dimensional vector would make a 3000-dimensional feature vector for a single image. That, too, only considers textual features. The DSSM-LSTM combination is used to reduce the individual vectors to 64-dimensional for increased processing speed.

Experimentation environment

Google Colab (Google Colab online link) is an online environment used to code and run python code in the browser, used for running all python codes for this paper. Colab gives free access to resources like GPUs essential for training deep learning models. The Colab notebooks use Google's cloud servers to execute code and require no configuration. This paper makes use of the Wang dataset (Wang, Li & Wiederhold, 2001), which contains 1000 images divided into ten classes, with 100 images per class. Each of these images is 224×224 in size and is colored. The dataset was split into a 70–30 train-test ratio for training the models.

The performance measure considered in this paper is accuracy.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)} \times 100 \quad (1)$$

Where TP = Count of True Positives, TN = Count of True Negatives, FP = Count of False Positives, FN = Count of False Negatives

Cosine similarity was used to calculate similarity among word vectors and obtain synonyms for each keyword assigned to an image.

$$similarity(A,B) = \frac{A \cdot B}{\| A \| \times \| B \|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2)$$

Results and discussion

Results for the paper are represented in this section. Table 1 shows the accuracy of each of the chosen classification algorithms to classify images solely based on their textual features like i) 300-dimensional word embedding for each keyword for an image obtained from word2-vec and ii) 64-dimensional word embedding obtained from subsequent DSSM-LSTM processing and addition of five synonyms for each keyword/ embedding. It is inferred from the comparison that the additional step of DSSM-LSTM and synonym addition for each algorithm increased the accuracy. The maximum increase in accuracy is observed using the Support Vector Machine (SVM) model, which saw an 8.33% increase in accuracy from 87.33% to 95.66%.

Pre-trained DCNN models were fine-tuned to obtain feature vectors for each image in the dataset. These feature vectors based solely on visual data were then fed to the classification algorithms mentioned below, and the accuracies are shown in Table 2. One can observe that the Xception model gave the best accuracy among each classification algorithm. Xception features fed to an SVM model provided the best accuracy from the set at 97.33%.

As discussed earlier, two fusion operations are performed in this paper. The first is the fusion of pre-trained model features (visual features) and 300-dimensional vectors for each keyword obtained from word2vec (textual features). The second fusion is of pre-trained model features (visual features) with the 64-dimensional vector obtained from DSSM-LSTM and synonym addition (additionally processed textual features). Similar to the results of Table 1, the additional steps of DSSM-LSTM and synonym addition resulted in increased accuracy overall. In these results, like in Table 2, Xception outperforms all other pre-trained models across all the classification algorithms. Also, similar to Table 2 results, the combination of Xception with SVM gave maximum accuracy of 98.33%. Also, it is observed that the fusion of visual features and textual features performed better than individually considering visual or textual features.

In Table 4, the results of the proposed method are compared with other modern CBIC methodologies. In some papers, if a particular feature set like textual features or fusion of textual and image features are not considered, they are left blank in the table. Therefore, one can infer that our proposed system gives the maximum accuracy of 98.33% among all the other methodologies. Also, if only visual features are compared, the proposed method gives the maximum accuracy of 97.33%. If only text-based features are considered, again, the proposed system gives the maximum accuracy of 95.66%.

Table 1
Comparison between the classification accuracies of word2vec-based Vectorized Text Descriptors and DSSM-LSTM-based reduced Vectorized Text Descriptors with synonyms on different classification algorithms.

Method	word2vec based embedding	DSSM-LSTM-based embedding with synonyms
Random Forest	93.33	95.33
SVM	87.33	95.66
Naive Bayes	85	86.66
Decision Trees	85.33	89

Table 2
Comparison of classification accuracies of feature vectors extracted from different pre-trained DeepCNN models after Transfer learning on different classification algorithms.

Method	Pre-trained Model	Auto-extracted features
Random Forest	Xception	96.66
	VGG	89.66
	MobileNet	95.33
	Inceptionv3	96
SVM	Xception	97.33
	VGG	89.66
	MobileNet	96.66
	Inceptionv3	95.33
Naive Bayes	Xception	96.33
	VGG	83.33
	MobileNet	93.66
	Inceptionv3	94.66
Decision Trees	Xception	94.66
	VGG	87.33
	MobileNet	91.66
	Inceptionv3	92.33

Table 3
Comparison of the classification accuracies of the two fusion vectors having visual and textual features on different classification algorithms.

Method	Model	Fusion of word2vec embeddings + CNN features	Fusion of DSSM-LSTM embedding with synonyms + CNN
Random Forest	Xception	97	97.66
	VGG	93.66	96.33
	MobileNet	95.66	97.33
	Inceptionv3	96.66	97
SVM	Xception	97.66	98.33
	VGG	92	96.66
	MobileNet	97	97.66
	Inceptionv3	96	97.33
Naive Bayes	Xception	97.66	98
	VGG	87	91.33
	MobileNet	94.33	96
	Inceptionv3	95	95.33
Decision Trees	Xception	95	96
	VGG	89.33	90.33
	MobileNet	92	93.33
	Inceptionv3	94	95.33

Table 4
Comparison of the proposed method with existing relevant techniques for Content-Based Image Classification.

Methodology	Visual feature accuracy	Text feature accuracy	Fusion accuracy
InceptionResNetv2 (Subhadip Maji et al., 2021)	96.12	–	–
Clustering-based Fusion Descriptor (Shikha, Pandove & Dahiya, 2020)	95	–	–
Semantic Segmentation (Ouni, Royer & Chevaldonné, 2021)	88	–	–
Re-ranking-based Hybrid approach (B. J. Dange, Yadav & Kshirsagar, 2020)	70	42	85
Low-level image features, the bag of words (Salahuddin Unar et al., 2019)	78	74	77
Proposed method	97.33	95.66	98.33

Conclusion

Both text-based and visual features, individually, perform satisfactorily but outperform when the fusion of these features is used for CBIC. Thus both features contribute important information for the

classification task, and their fusion is essential to obtain the best performance for the CBIC task. The proposed model gets the text descriptors from Google Cloud Vision API. These labels are replaced by 300-dimensional numerical vectors obtained from word2vec. The paper proposed a novel technique for improved accuracy using textual features. The 300-dimensional word embedding was reduced to 64-dimensional embedding after training on DSSM-LSTM. Each keyword for an image was then replaced with five synonyms by ranking the new 64-dimensional word embeddings by cosine similarity. It is observed that these additional steps provided increased accuracy. Pre-trained models are extensively used today for feature extraction. This paper used four fine-tuned pre-trained models for our subtask, namely, VGG16, MobileNet, Inceptionv3, and Xception. Here four classification algorithms are used for performing the classification task using our feature vectors. The algorithms were Random Forest, SVM, Naive Bayes, and Decision Trees. Finally, it is observed that the fusion of vectorized text descriptors using the 3-step embedding process and auto-extracted features from pre-trained models gave the highest performance.

Funding

No funding was received for this work.

CRediT authorship contribution statement

Sudeep D. Thepade: Conceptualization, Methodology, Writing – original draft, Supervision. **Jovian A. Jaison:** Data curation, Investigation, Visualization, Software, Validation.

Conflicts of Interest

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

Howard Andrew, Menglong, Zhu, Bo, Chen, Dmitry, Kalenichenko, Weijun, Wang, Tobias, Weyand … Hartwig, Adam (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. 2017.

Barbhuiya, A. A., Karsh, R. K., & Jain, R. (2021). CNN based feature extraction and classification for sign language. *Multimedia tools and applications, 80*, 3051–3069. https://doi.org/10.1007/s11042-020-09829-y, 2021.

Bhandi, V., & Sumithra Devi, K. A. (2019). Image Retrieval by Fusion of Features from Pre-trained Deep Convolution Neural Networks. In *1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)* (pp. 35–40). https://doi.org/10.1109/ICATIECE45860.2019.9063814, 2019.

Chollet, Francois. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. 1800–1807. doi:10.1109/CVPR.2017.195.

Szegedy Christian , Vincent, Vanhoucke, Sergey, Ioffe, Jon, Shlens, & Wojna, Z.B. (.2016). Rethinking the Inception Architecture for Computer Vision. CVPR 2017. doi:10.1109/CVPR.2016.308.

Dange, B. J., Yadav, S. K., & Kshirsagar, D. B. (2020). Enhancing Image Retrieval and *Re*-ranking Efficiency using Hybrid approach. In *International Conference on Smart Innovations in Design, Environment, Management, Planning, and Computing (ICSIDEMPC 2020)* (pp. 20–26). https://doi.org/10.1109/ICSIDEMPC49020.2020.9299579

Google Cloud Vision API. online link. https://cloud.google.com/vision/(Last accessed: 18 October (2021)).

Google Colab. Online link https://colab.research.google.com/?utm_source=scs-index (Last accessed: 18 October (2021)).

Google word2vec. Online link.https://code.google.com/archive/p/word2vec/(Last accessed: 18 October (2021)).

Hamid, Palangi, Deng, Li, Shen, Yelong, Gao, Jianfeng, He, Xiaodong, Chen, Jianshu, … Ward, Rabab (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio Speech and Language Processing, 24*(4), 694–707. https://doi.org/10.1109/TASLP.2016.2520371. Https://doi.org/ April 2016.

Huang, Xiao, Wang, Cuizhen, Li, Zhenlong, & Ning, Huan (2019). A visual–textual fused approach to automated tagging of flood-related tweets during a flood event. *International Journal of Digital Earth, 12*(11), 1248–1264. https://doi.org/10.1080/17538947.2018.1523956

Inceptionv3 Image reference. https://cloud.google.com/tpu/docs/inception-v3-advanced (Last accessed: 11 January (2022)).

Li, Jia, & Wang, James Z. (2003). Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*(9), 1075–1088. 2003Dataset available at http://wang.ist.psu.edu/docs/related/.

Li, Z., Nie, F., Chang, X., Yang, Y., Zhang, C., & Sebe, N. (2018a). Dynamic Affinity Graph Construction for Spectral Clustering Using Multiple Features. *IEEE transactions on neural networks and learning systems, 29*(12), 6323–6332. https://doi.org/10.1109/TNNLS.2018.2829867, 2018 DecEpub 2018 May 18. PMID: 29994548.

Li, Z., Nie, F., Chang, X., Nie, L., Zhang, H., & Yang, Y. (2018b). Rank-Constrained Spectral Clustering With Flexible Embedding. *IEEE Transactions on Neural Networks and Learning Systems, 29*(12), 6073–6082. https://doi.org/10.1109/TNNLS.2018.2817538. Dec. 2018.

Li, Z., Yao, L., Chang, X., Zhan, K., Sun, J., & Zhang, H. (2019). Zero-shot event detection via event-adaptive concept relevance mining. *Pattern Recognition, 88*, 595–603. https://doi.org/10.1016/j.patcog.2018.12.010, 2019.

Mahalakshmi, P., & Fatima, N. S. (2020). Collaborative text and image based information retrieval model using BiLSTM and residual networks. In *3rd International Conference on Intelligent Sustainable Systems (ICISS)* (pp. 958–964). https://doi.org/10.1109/ICISS49785.2020.9315886, 2020.

Messina, N., Amato, G., Carrara, F., et al. (2020). Learning visual features for relational CBIR. *International Journal of Multimedia Information Retrieval, 9*, 113–124. https://doi.org/10.1007/s13735-019-00178-7, 2020.

MobileNet Image reference. https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470 (Last accessed: 11 January (2022)).

Ouni, A., Royer, E., Chevaldonné, M., et al. (2021). Leveraging semantic segmentation for hybrid image retrieval methods. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-021-06087-3, 2021.

Shikha, B., Pandove, Gitanjali, & Dahiya, Pawan (2020). A genesis of an effective clustering-based fusion descriptor for an image retrieval system. 2020. doi:10.1007/978-981-15-4992-2_29.

Simonyan, Karen, & Zisserman, Andrew. (2014). Very deep convolutional networks for large-scale image recognition. 2014. arXiv 1409.1556.

Subhadip, Maji, & Smarajit, Bose (2021). CBIR using features derived by deep learning. *ACM/IMS Transactions on Data Science, 2*(3), 24. https://doi.org/10.1145/3470568. Https://doi.org/ Article 26August 2021pages.

Thepade, S. D., & Dindorkar, M. R. (2022). Fusing deep convolutional neural network features with Thepade's SBTC for land usage identification. *Engineering Science and Technology, An International Journal, 27*. https://doi.org/10.1016/j.jestch.2021.05.018. Volume2022.

Salahuddin, Unar, Wang, Xingyuan, Wang, Chunpeng, & Wang, Yu (2019). A decisive content based image retrieval approach for feature fusion in visual and textual images. *Knowledge-Based Systems, 179*, 8–20. https://doi.org/10.1016/j.knosys.2019.05.00. Volume2019PagesISSN 0950-7051.

VGG16 Image Reference. https://medium.com/analytics-vidhya/invoice-information-extraction-using-ocr-and-deep-learning-b79464f54d69 (Last accessed: 11 January (2022)).

Wang, James Z., Li, Jia, & Wiederhold, Gio (2001). SIMPLIcity: Semantics-sensitive Integrated Matching for Picture Libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 23*(9), 947–963. no2001Dataset available at http://wang.ist.psu.edu/docs/related/.

Xception Image reference. https://arxiv.org/abs/1610.02357 (Last accessed: 11 January (2022)).