A2_2021013_Report

Image Preprocessing

Image preprocessing is an essential step in computer vision tasks, as it helps to normalize and enhance the input images, making them more suitable for further processing and analysis. In this project, we applied several preprocessing techniques to the input images to improve the performance of our image similarity search algorithm. The preprocessing steps are as follows:

1. **Contrast Adjustment**: We adjusted the contrast of the images using the `ImageEnhance.Contrast` function from the Python Imaging Library (PIL). This step helps to improve the visibility of image details by increasing the difference between light and dark regions. The contrast was enhanced by applying a contrast factor of 1.5.
2. **Resizing**: The input images were resized to a fixed size of 224 × 224 pixels using the `img.resize` function from PIL. Resizing the images to a consistent size is crucial for feeding them into deep learning models or other computer vision algorithms that expect a specific input size.
3. **Geometric Transformations**: We applied a mirroring operation to the images using the `ImageOps.mirror` function from PIL. Mirroring introduces additional variations in the input data, which can help improve the robustness of the image similarity search algorithm.
4. **Random Flips**: To further augment the input data, we randomly flipped the images horizontally with a 50% probability using the `img.transpose(Image.FLIP_LEFT_RIGHT)` function from PIL. Random flips introduce additional variations in the input data, which can help to improve the generalization performance of the image similarity search algorithm.
5. **Brightness Adjustment**: We adjusted the brightness of the images using the `ImageEnhance.Brightness` function from PIL. Brightness adjustment can help to compensate for variations in lighting conditions and improve the visibility of image details. The brightness was enhanced by applying a brightness factor of 1.2.

6. **Exposure Adjustment**: We adjusted the exposure of the images using the `ImageEnhance.Color` function from PIL. Exposure adjustment can help to compensate for variations in lighting conditions and improve the visibility of image details. The exposure was adjusted by applying an exposure factor of 0.8.

MobileNet for Feature Extraction

In this project, we utilized a pre-trained MobileNet model for extracting relevant features from the input images. MobileNet is a lightweight and efficient convolutional neural network (CNN) architecture designed for mobile and embedded vision applications. By leveraging the knowledge learned by the MobileNet model on a large-scale dataset like ImageNet, we can effectively transfer this knowledge to our image similarity search task.

The implementation of the MobileNet model for feature extraction can be broken down into the following steps:

1. **Loading the Pre-trained MobileNet Model**: We loaded the pre-trained MobileNet model weights from the ImageNet dataset using the `MobileNet` function from the Keras library. Specifically, we used the following parameters:

```
base_model = MobileNet(weights='imagenet', include_top=False,
```

2. **Preprocessing the Input Images**: Before feeding the images into the MobileNet model, we preprocessed them using the `preprocess_input` function from Keras. This step ensures that the input images are properly normalized and scaled according to the requirements of the pre-trained model.
3. **Extracting Features using the Pre-trained MobileNet Model**: To extract features from the preprocessed images, we created a new model by removing the top layers from the pre-trained MobileNet model. We used the `global_average_pooling2d` layer as the output layer, which provides a feature vector of a fixed size for each input image.

```
pythonCopy codefeature_extractor = Model(inputs=base_model.in
put, outputs=base_model.get_layer('global_average_pooling2
d').output)
features = feature_extractor.predict(preprocessed_images)
```

The `feature_extractor` model takes the input images and outputs the feature vectors from the `global_average_pooling2d` layer. We then used the `predict` method to extract the features from the preprocessed images, resulting in a feature matrix `features` containing the feature vectors for all input images.

By utilizing the pre-trained MobileNet model for feature extraction, we leverage the powerful feature representations learned by the model on the large-scale ImageNet dataset. These extracted features capture essential visual characteristics of the input images, such as textures, shapes, and patterns, which are crucial for accurate image similarity search.

The extracted features from the MobileNet model serve as input to the subsequent stages of our image similarity search pipeline, where we compute the similarity between the input image and other images in the dataset based on these feature representations.

Using Cosine Similarity for Image and Text Retrieval:

In the first implementation, we utilized cosine similarity as a measure of similarity between feature vectors representing images and text reviews. Cosine similarity is a widely used metric to determine the similarity between two vectors in a multi-dimensional space. It measures the cosine of the angle between two vectors and provides a value between -1 and 1, where 1 indicates perfect similarity and -1 indicates perfect dissimilarity.

For image retrieval, we first extracted features from each image using a pre-trained neural network model (e.g., MobileNet). These features represent high-level visual characteristics of the images. Then, given an input image, we computed the cosine similarity between its feature vector and the feature vectors of all other images in the dataset. The images with the highest cosine similarity scores were considered the most similar to the input image.

Similarly, for text retrieval, we used TF-IDF (Term Frequency-Inverse Document Frequency) vectors to represent the text reviews. TF-IDF is a numerical statistic that reflects the importance of a word in a document relative to a collection of documents. We calculated the cosine similarity between the TF-IDF vector of the input text review and the TF-IDF vectors of all other reviews in the dataset. The reviews with the highest cosine similarity scores were considered the most similar to the input text review.

Using Composite Similarity for Combined Retrieval:

In the second case, we aimed to combine the results from image and text retrieval to provide a more comprehensive similarity measure. To achieve this, we introduced the concept of composite similarity. The composite similarity score was calculated by taking the average of the cosine similarity scores obtained from image and text retrieval.

For each pair of similar images and reviews identified through image and text retrieval separately, we computed composite similarity scores by averaging the cosine similarity scores obtained for images and reviews. This approach allowed us to incorporate both visual and textual information and provide a holistic measure of similarity between the input image and text review pair and other images and reviews in the dataset.

By combining the results of image and text retrieval using composite similarity, we aimed to enhance the accuracy and robustness of similarity assessment, thereby improving the effectiveness of content-based recommendation systems and information retrieval applications.