|  |  |
| --- | --- |
| Ex No: 5  Date: 04/09/2024 | Transfer Learning for Fish and Flower Classification using Convolutional Neural Networks. |

**Objective:**

This project demonstrates the use of transfer learning to build and fine-tune a Convolutional Neural Network (CNN) for classifying images of flowers. The primary goal is to leverage a pre-trained model (Inception V3) and adapt it to recognize and classify various types of flowers from a new dataset. The process includes preprocessing the dataset, scaling the images, fine-tuning the model, and evaluating its performance on unseen test data. The objective is to achieve high accuracy in flower classification by utilizing the powerful feature extraction capabilities of a pre-trained deep learning model.

**Variation Used:** Inception V3

**Code Explanation:**

* **Fish Classification**
* The first code cell installs the tensorflow\_hub package, which is essential for loading pre-trained models from TensorFlow Hub. The output confirms that both tensorflow\_hub and tf-keras were installed successfully.
* This section of code installs the tf\_keras package, ensuring that the necessary TensorFlow and Keras packages are available.
* In this code block, the MobileNetV2 model is commented out, and the InceptionV3 model is selected as the classifier by setting the classifier\_model variable to the InceptionV3 URL on TensorFlow Hub.
* The code defines the image shape as (224, 224) and loads the pre-trained InceptionV3 model from TensorFlow Hub using hub.KerasLayer. The model's weights are frozen (trainable=False) to prevent any updates during training.
* The pre-trained model is incorporated into a Sequential model, with the first layer being a Lambda layer that calls the pre-trained model.
* The code loads an image of a goldfish using tf.image.decode\_jpeg, resizes it to the defined IMAGE\_SHAPE, and expands its dimensions to match the model's expected input shape (including batch size).
* The code uses the pre-trained InceptionV3 model to predict the class of the goldfish image (classifier.predict). The output is a vector with 1,001 elements, where each element represents the logit (confidence) for one of the classes in the ImageNet dataset.
* The index of the predicted class (the class with the highest logit value) is determined using np.argmax. The predicted\_label\_index gives the index of the class with the highest predicted probability.
* This code snippet downloads a file called ImageNetLabels.txt from a given URL, which contains the class labels for the ImageNet dataset, with each label on a separate line. The with open statement opens the file, reads all the lines, and stores them as a list in the image\_labels variable. The code then prints the first five labels from this list to confirm that the labels were loaded correctly.
* Finally, the code retrieves the label corresponding to the predicted class index (predicted\_label\_index) from the image\_labels list. In this example, it returns the label "goldfish," indicating that the model correctly classified the image as a goldfish.
* **Flower Classification**
* The code uses TensorFlow's utility function tf.keras.utils.get\_file to download a compressed dataset of flower images from a specified URL (dataset\_url). The dataset is saved in the current directory (cache\_dir='.'), and the untar=True option automatically extracts the contents.
* The downloaded dataset's directory path (data\_dir) is converted into a pathlib.Path object for easier file management. The code then lists the first five .jpg images in the dataset by using glob to search through the directory and its subdirectories.
* The total number of .jpg images in the dataset is counted (image\_count), and this count is printed. Additionally, the code lists image files specifically from the "roses" and "tulips" subdirectories, displaying the first few images from each category using the PIL.Image.open function.
* A dictionary (flowers\_images\_dict) is created, where each key represents a flower category (e.g., "roses," "daisy") and the corresponding value is a list of image file paths for that category.
* Another dictionary (flowers\_labels\_dict) is created to map each flower category to a unique numeric label. The first five image paths from the "roses" category are retrieved and displayed.
* The code processes flower images for machine learning tasks. It reads images from the flowers\_images\_dict, resizes each to 224x224 pixels, and adds the resized images to the list X. Corresponding labels from flowers\_labels\_dict are stored in the list y. If an image can't be read, an error message is printed. Finally, X and y are converted into NumPy arrays for further use.
* The images are prepared and scaled for prediction with a pre-trained model. The dataset is first split into training and testing sets using train\_test\_split. Images in both sets are then normalized by dividing pixel values by 255.
* The first three images in X are resized to match the IMAGE\_SHAPE expected by the model, and the first image is displayed without axis labels using Matplotlib's imshow.
* A pre-trained model is used to make predictions and fine-tune it for a new flower image classification task. The model predicts the labels of three resized flower images using a classifier, with the most likely class extracted using np.argmax.
* Next, the Inception V3 model, pre-trained on TensorFlow Hub, is loaded without its top layer, allowing it to be fine-tuned on a new dataset. This pre-trained model is added as a feature extractor in a new Sequential model, followed by a dense layer to output predictions for five flower classes.
* The code ensures that the parameters of the pre-trained model are not trainable and then prints the model summary for an overview of the layers and parameters.
* The code fine-tunes a pre-trained model on a flower image dataset and makes predictions on new images. The model is compiled using the Adam optimizer, sparse categorical cross-entropy loss, and accuracy as a metric.
* It is trained for five epochs on the scaled training data, and its performance is evaluated on the test data. A helper function preprocess\_image is defined to load and preprocess a new image by resizing it to 224x224 pixels, normalizing the pixel values, and expanding the dimensions to fit the model input.
* The model predicts the class of a test image (sun.jpg), and the predicted class index is identified using np.argmax. Finally, the predicted class is matched to the corresponding flower name from a predefined list and printed.

**GitHub Link:**

https://github.com/ManeshaMadhu/DeepLearning-5thsem-/tree/main/Lab%205