|  |  |
| --- | --- |
| 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 necessary to load pre-trained models from TensorFlow Hub. The output confirms the successful installation of tensorflow\_hub and tf-keras.
  + The code here shows the installation of the tf\_keras package, ensuring that the required TensorFlow and Keras packages are available.
  + The code block comments out MobileNetV2 and selects the InceptionV3 model as the classifier model 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 them from being updated during training.
  + The pre-trained model is wrapped 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 the tf.image.decode\_jpeg method, resizes it to the specified IMAGE\_SHAPE, and expands its dimensions to match the expected input shape of the model (batch size included).
  + The code predicts the class of the goldfish image using the pre-trained InceptionV3 model (classifier.predict). The result is a vector with 1,001 elements, where each element corresponds to the logit (confidence) for one of the classes in the ImageNet dataset.
  + Finds the index of the predicted class (the class with the highest logit value) using np.argmax. The predicted\_label\_index gives the index of the class with the highest predicted probability.
  + The code snippet is downloading a file called `ImageNetLabels.txt` from a given URL, which contains the class labels for the ImageNet dataset, with one label per line. The `with open` statement opens the file, reads all the lines, and stores them as a list in the `image\_labels` variable. Finally, the code prints the first five labels from this list to verify that the labels have been loaded correctly.
  + The code retrieves the label corresponding to the predicted class index (`predicted\_label\_index`) from the `image\_labels` list. In this case, it returns the label "goldfish," which indicates that the model has classified the image as a goldfish.
* **Flower Classification**
* The code downloads a compressed dataset of flower photos from the specified URL (`dataset\_url`) using TensorFlow's utility function `tf.keras.utils.get\_file`. The dataset is downloaded to the current directory (`cache\_dir='.'`), and the `untar=True` option automatically extracts the contents of the compressed file.
* The code converts the downloaded dataset directory path (`data\_dir`) into a `pathlib.Path` object for easier file handling. It then lists the first five `.jpg` image files within the dataset by using `glob` to search through the directory and subdirectories.
* The code counts the total number of `.jpg` images in the dataset (`image\_count`) and prints the result. It then lists the image files specifically in the "roses" and "tulips" subdirectories, displaying the first few images from each category using the `PIL.Image.open` function to open and view them.
* The code creates a dictionary (`flowers\_images\_dict`) where each key is a flower category (e.g., "roses," "daisy") and the corresponding value is a list of image file paths for that category.
* The code also creates a second dictionary (`flowers\_labels\_dict`) that maps each flower category to a unique numeric label. The last line retrieves and displays the first five image paths from the "roses" category.
* The code processes images of flowers and prepares them for machine learning tasks. It reads images from a dictionary `flowers\_images\_dict`, resizes each image to 224x224 pixels, and appends the resized images to the list `X`.
* The corresponding labels for each flower, retrieved from `flowers\_labels\_dict`, are stored in the list `y`. The code handles errors by printing a message if an image cannot be read. Finally, it converts the lists `X` and `y` into NumPy arrays for further use.
* The code prepares and scales images for prediction using a pre-trained model. It first splits the dataset into training and testing sets using `train\_test\_split`. The images in both sets are then scaled by dividing pixel values by 255 to normalize them.
* The code resizes the first three images in `X` to match the `IMAGE\_SHAPE`, which is expected by the model. Finally, it displays the first image in the dataset without axis labels using Matplotlib's `imshow`.
* The code is using a pre-trained model to make predictions and then fine-tune it for a new flower image classification task. First, it predicts the labels of three resized flower images using a classifier and extracts the most likely class using `np.argmax`.
* Then, it loads a pre-trained Inception V3 model from TensorFlow Hub, excluding the top layer, so the model can be fine-tuned on a new dataset. The pre-trained model is added as a feature extractor in a new `Sequential` model, followed by a dense layer that will output predictions for five flower classes. The code checks the parameters of the pre-trained model to ensure they aren't trainable.
* Lastly, it prints the model summary to provide 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. First, it compiles the model using the Adam optimizer, a sparse categorical cross-entropy loss, and accuracy as a metric.
* The model is then trained for five epochs on the scaled training data. After training, the model's performance is evaluated on the test data. A helper function `preprocess\_image` is defined to load and preprocess a new image, 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/spoorthytorne/fundamentals-of-Deep-learning/tree/main/Lab%205