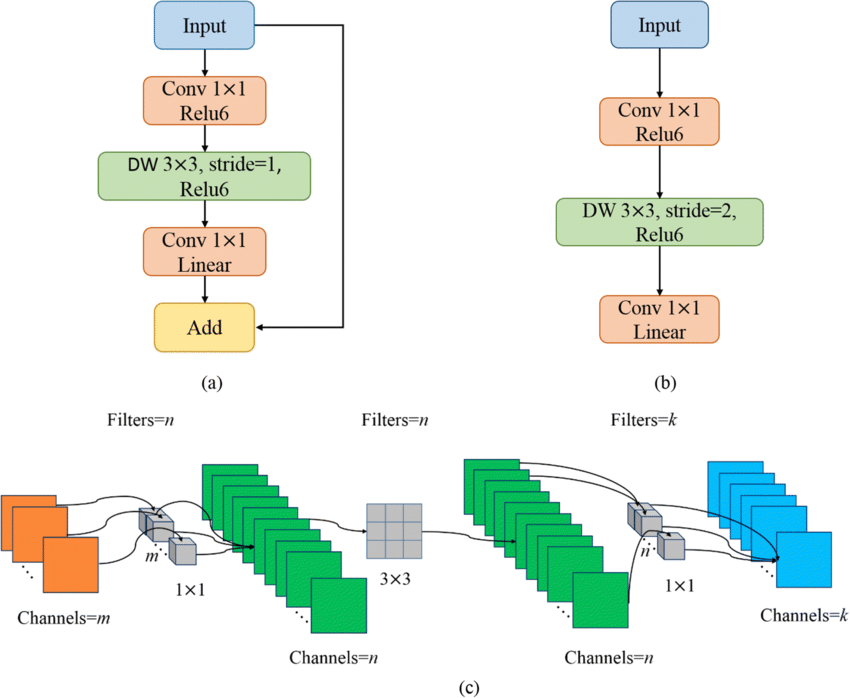
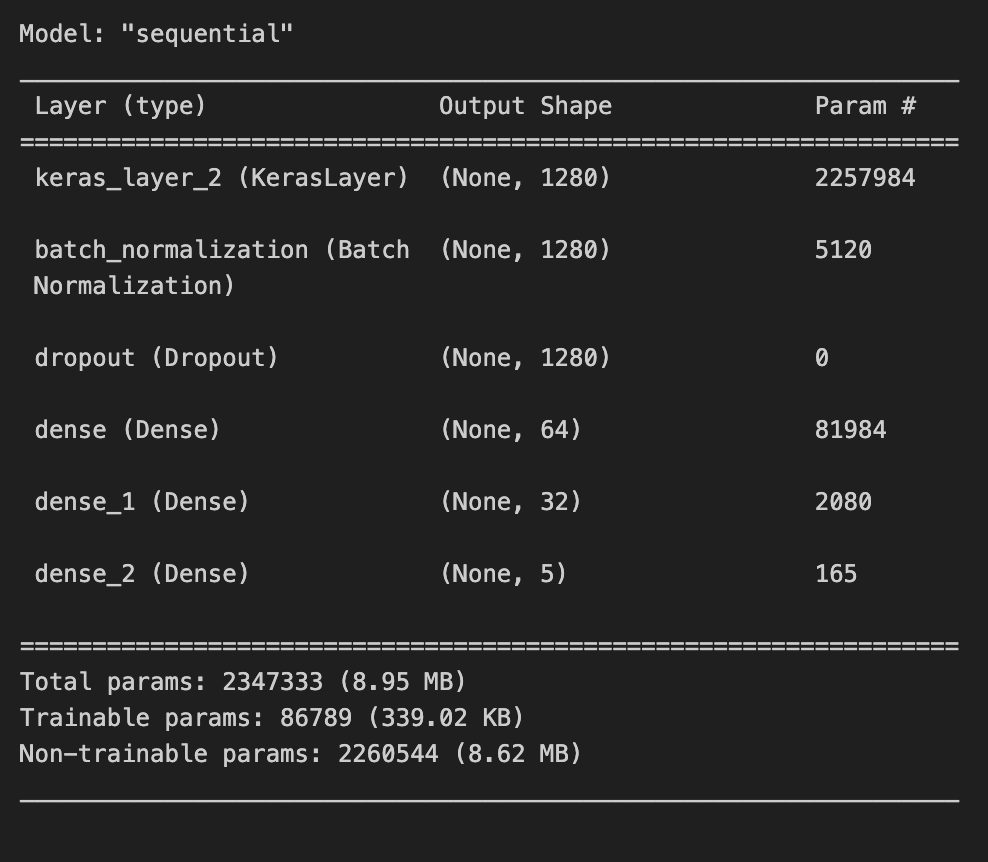
|  |  |
| --- | --- |
| **Ex No: 4**  **Date: 18 Sep** | **Transfer Learning in Image Classification** |

**Objective:**The objective is to apply transfer learning for classifying images of five distinct flower types. This involves utilizing a pre-trained model, such as MobileNetV2 from TensorFlow Hub, and fine-tuning it on a flower dataset. By leveraging the pre-trained features of MobileNetV2, the task aims to significantly reduce the time and computational resources required for training compared to building a model from scratch. The pre-trained model will be re-trained on a specific classification problem, focusing on five flower categories: daisy, dandelion, rose, sunflower, and tulip. The approach ensures faster convergence and improved performance by utilizing the knowledge embedded in the pre-trained model’s weights.

**Description:**

Transfer learning is a machine learning technique where a model trained for one task is repurposed to address a related task with a smaller dataset. This approach leverages the knowledge the model has already learned from large datasets, reducing the need for extensive data and computational resources. In image classification, for example, pre-trained models like those trained on ImageNet can extract useful features from images and be fine-tuned to classify new categories. This process accelerates training, improves accuracy, and helps avoid overfitting when limited data is available for the new task.

MobileNetV2 is a lightweight and efficient deep learning model architecture designed specifically for mobile and edge devices. It uses depthwise separable convolutions to reduce the number of parameters and computational demands while maintaining high performance. The model also features inverted residual blocks and linear bottlenecks, which facilitate the flow of information through the network and minimize the feature space size. Pre-trained on large datasets like ImageNet, MobileNetV2 can easily be adapted for transfer learning tasks, such as flower classification, by replacing its top layer with a new classifier tailored to the specific dataset. Its efficiency and compact design make it an ideal choice for tasks requiring fast and resource-efficient computation.

**Model Summary:**

**Fig1:** A sequential CNN model used in Transfer Learning.

**Building The Parts Of The Algorithm**

**1. Imports and Installation:**

* The key libraries involved in building the algorithm include TensorFlow, Keras, TensorFlow Hub, OpenCV, and Matplotlib.
* TensorFlow and Keras are used for constructing and training the deep learning model.
* TensorFlow Hub is used for loading the pre-trained MobileNetV2 model for transfer learning.
* OpenCV is used for handling image preprocessing tasks such as resizing and normalization.
* Matplotlib is used for visualizing training accuracy, loss curves, and results.

**2. Pre-trained Model - MobileNetV2 :**

* A pre-trained MobileNetV2 model is loaded using hub.KerasLayer. This model is designed for efficient image classification tasks and acts as a feature extractor.
* The classification head (top layers) of the MobileNetV2 is removed, and it is replaced by a custom classifier to adapt the model for flower classification.
* MobileNetV2 offers a balance between model size and accuracy, making it suitable for real-time classification tasks.

**3. Model Construction:**

* **Pre-trained Feature Extractor:**
  + The model starts with the pre-trained MobileNetV2 layer (keras\_layer\_2), which outputs a feature map of shape (None, 1280) (indicating 1280 features).
* **Batch Normalization:**
  + A batch\_normalization layer is applied after the feature extractor to normalize the activations and speed up convergence.
* **Dropout:**
  + A dropout layer is added to prevent overfitting by randomly deactivating a fraction of neurons during training.
* **Dense Layers:**

1. A dense layer with 64 neurons (dense) is added with trainable parameters, making the model more flexible for flower classification.
2. Another dense layer with 32 neurons (dense\_1) refines the representation further.
3. A final dense layer (dense\_2) with 5 output neurons corresponds to the five flower categories (as indicated by the small number of parameters in the final layer).

* **Output Layer:**
  + The output layer uses 5 neurons, indicating that the model is set for a 5-class flower classification task.

**4. Model Compilation and Training:**

* **Compilation:**
  + The model is compiled with the Adam optimizer, known for its adaptability in learning rates, to minimize the loss and improve accuracy.
  + The loss function used is Sparse Categorical Crossentropy, which is ideal for multi-class classification tasks.
* **Training:**
  + The model is trained on the flower classification dataset using 10 epochs, a batch size of 64, and a 20% validation split.
  + Data augmentation and preprocessing techniques can also be employed before feeding data into the model.

**5. Evaluation and Prediction:**

* Model Evaluation: After training, the model’s performance is evaluated using the test dataset to assess accuracy and other relevant metrics.
* Prediction: A function is implemented to load an image, preprocess it (resize, normalize, etc.), and predict the flower class using the trained model.
  + The model predicts one of the five flower categories based on the learned features.

**Conclusion:**

The transfer learning approach used in this assignment demonstrates the efficiency of leveraging pre-trained models, such as MobileNetV2, for specific tasks like flower classification. By utilizing a model already trained on a large dataset (ImageNet), the computational cost and training time are greatly reduced. The model extracts general features from the pre-trained data and is then fine-tuned to perform well on the new, smaller dataset with minimal additional training. This approach highlights the effectiveness of transfer learning in efficiently handling small datasets and complex tasks, making it a valuable tool in image classification and broader machine learning applications.

**Github:** https://github.com/Kashishvarmaa/DL-CS3232/tree/main/Lab\_4