PROGRAM 5

AIM-Introduction to Transfer Learning and Fine Tuning in Deep CNN. Also, discuss various Pre-trained networks with their advantage and drawbacks.

THEORY-

1. Introduction to Transfer Learning: Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a second task. In deep learning, particularly with Convolutional Neural Networks (CNNs), transfer learning has proven highly effective because CNNs trained on large datasets like ImageNet have already learned useful features such as edge detection, shape recognition, and texture identification. These features can be generalized across many different types of image recognition tasks.

Instead of training a deep learning model from scratch (which can be computationally expensive and time-consuming, requiring large datasets), transfer learning allows us to leverage the knowledge learned by a pre-trained model.

2. Concept of Transfer Learning in Deep CNN: In deep CNNs, lower layers learn general features like edges and textures, while higher layers learn more task-specific features. In transfer learning, we use the lower layers of a pre-trained network as a feature extractor and modify the higher layers to suit the new task.

Transfer learning in CNNs typically follows these steps:

- **Using a Pre-trained Model:** Load a model that has already been trained on a large dataset, like ImageNet.
- **Modifying the Architecture:** Replace the output layer (classifier) of the model to suit the new dataset's classes.
- **Freezing Some Layers:** Freeze the initial layers of the model to retain the learned features.
- **Training on New Data:** Train the remaining layers and the new classifier on the new dataset.

This significantly reduces training time and provides better performance, especially when the new dataset is smaller.

3. Fine-Tuning: Fine-tuning is a specific type of transfer learning where you not only replace the output layer but also allow some (or all) of the pre-trained model's layers to be updated during training.

In typical transfer learning:

- The pre-trained model's layers are frozen (i.e., their weights are not updated).
- Only the newly added classifier (top layer) is trained on the new dataset.

In fine-tuning:

- Some of the pre-trained layers (usually higher layers closer to the output) are unfrozen, meaning they can learn and adjust weights based on the new dataset.
- The learning rate during fine-tuning is usually lower to ensure the changes in the weights are small and don't drastically affect the model's pre-learned features.

4. Types of Transfer Learning: There are two primary types of transfer learning in deep learning:

a. Feature Extraction (Frozen Layers):

- Here, the pre-trained model is used as a feature extractor. The layers are frozen, and the output layer is modified to suit the new dataset. Only the new classifier is trained on the new data.
- This method works well when the dataset is small and the new task is similar to the one the model was pre-trained on.

b. Fine-Tuning (Unfreezing Layers):

- Fine-tuning involves unfreezing some of the layers of the pre-trained model, allowing them to learn from the new dataset.
- Fine-tuning is typically used when the dataset is larger, or the new task is somewhat different from the original task.
- It allows the model to adapt the learned representations to the specifics of the new task.

5. Common Pre-trained Networks:

Several CNN architectures are widely used in transfer learning and fine-tuning. These networks are typically trained on large datasets like ImageNet, which contains over 1 million images and 1000 classes.

a. VGG (Visual Geometry Group) Networks:

- VGG16 and VGG19 are some of the most popular CNN architectures. They have 16 and 19 layers, respectively.
- They use very simple and uniform architecture, consisting of 3×3 convolutional layers followed by max-pooling.
- **Advantages:** VGG networks are straightforward, with consistent layer structures.
- **Disadvantages:** They are computationally expensive, requiring a lot of memory and processing power.

b. ResNet (Residual Networks):

- ResNet solves the vanishing gradient problem by introducing residual connections, which allow gradients to flow more easily through the network.
- ResNet models come in various sizes, such as ResNet-50, ResNet-101, etc., indicating the depth of the model.

- **Advantages:** Allows for much deeper networks while maintaining good performance. It generalizes well and achieves high accuracy.
- **Disadvantages:** ResNet models are larger and require more computational resources than simpler architectures.

c. Inception (GoogLeNet):

- InceptionNet uses different-sized convolutional filters in the same layer, allowing the network to capture multi-scale features in the input.
- Inception models (e.g., Inception-V3) are computationally efficient, using fewer parameters.
- **Advantages:** High accuracy with reduced computational requirements.
- **Disadvantages:** The architecture is more complex to implement and fine-tune.

d. MobileNet:

- MobileNet is designed for mobile and embedded applications where computational resources are limited.
- It uses depthwise separable convolutions, significantly reducing the number of parameters and computations required.
- **Advantages:** Low computational cost, small size, and optimized for real-time applications.
- **Disadvantages:** Slightly lower accuracy compared to larger models like ResNet or Inception.

6. Advantages of Transfer Learning:

- **Reduced Training Time:** Since the base layers are already pre-trained, training time is significantly reduced.
- **Better Performance with Less Data:** Transfer learning allows better generalization, even with smaller datasets.
- **Overcoming the Need for Large Datasets:** Deep learning models require vast amounts of labeled data for training, which may not always be available. Transfer learning allows you to apply models that were trained on large datasets to your own smaller datasets.

7. Drawbacks of Transfer Learning:

- **Domain Specificity:** If the task or domain of the new dataset is very different from the original dataset (e.g., medical images vs. natural images), transfer learning may not work well.
- **Large Models:** Many pre-trained networks are quite large, requiring more computational resources, even if you're only using them as a feature extractor.

- **Overfitting:** In some cases, transferring too much knowledge from the pre-trained model can lead to overfitting, especially when the dataset is small.

8. Applications of Transfer Learning:

a. Image Classification: Transfer learning is frequently used in image classification tasks where models like VGG, ResNet, and Inception have shown excellent performance.

b. Object Detection and Segmentation: Pre-trained models can be fine-tuned for tasks like detecting objects in an image or segmenting images into different classes.

c. Natural Language Processing (NLP): Transfer learning is also widely used in NLP tasks where models like BERT and GPT-3 are pre-trained on vast corpora of text and fine-tuned for specific language-related tasks.

d. Medical Imaging: Transfer learning has shown promise in medical imaging tasks like disease diagnosis, where data is scarce but critical.

VIVA QUESTIONS

1. **What is transfer learning?**
    - Transfer learning refers to using a model trained on one task as the starting point for a model on another related task.
2. **Why do we use transfer learning in CNNs?**
    - It is used to leverage the learned features from large datasets to reduce training time and improve performance on tasks with smaller datasets.
3. **What is fine-tuning in CNNs?**
    - Fine-tuning is the process of retraining some or all layers of a pre-trained network on a new dataset to adapt it to the new task.
4. **What is the difference between transfer learning and fine-tuning?**
    - In transfer learning, the pre-trained model is used without changing the weights. In fine-tuning, we adjust some or all of the weights based on the new dataset.
5. **Give an example of a pre-trained CNN model.**
    - Examples include VGG16, ResNet, InceptionNet.

PROGRAM-6

AIM - Implementation of Transfer Learning using Pre-trained Model (MobileNetV2) for Image Classification in Python

THEORY-

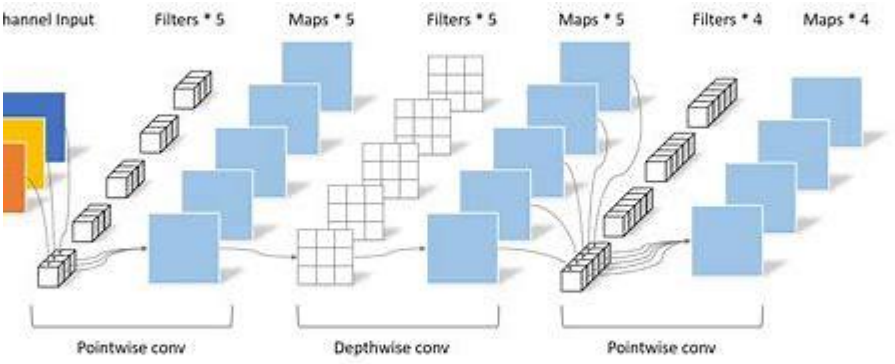
Introduction to Transfer Learning:

Transfer learning is a machine learning technique where a model trained on one task is repurposed to solve another related task. In deep learning, especially with Convolutional Neural Networks (CNNs), transfer learning is commonly used to leverage models pre-trained on large datasets such as ImageNet for smaller and more specific tasks. This saves time and computational resources and can often result in higher accuracy than training a model from scratch, particularly when the new task has limited data.

MobileNetV2:

MobileNetV2 is a lightweight deep neural network architecture designed by Google for mobile and embedded vision applications. It is optimized to perform well even on devices with limited computational power, such as smartphones. The architecture is based on **depthwise separable convolutions**, which significantly reduce the number of parameters and computational cost while retaining model accuracy.

MobileNetV2 is particularly suitable for transfer learning and fine-tuning because it balances performance and efficiency, making it an excellent choice for real-time applications or resource-constrained environments.



MobileNetV2 Architecture:

MobileNetV2 builds on its predecessor, MobileNetV1, and introduces several key improvements:

- **Depthwise Separable Convolutions:** This technique separates convolution into two operations:
    - **Depthwise convolution:** Applies a single filter to each input channel.
    - **Pointwise convolution:** Applies a 1x1 convolution to combine the outputs of the depthwise convolution. This reduces the number of computations and parameters, making the model lighter.

- **Inverted Residuals and Linear Bottlenecks:** In traditional residual networks (like ResNet), the residual blocks allow the input to bypass layers and directly connect to the output. In MobileNetV2, these residual connections are "inverted" and involve a linear bottleneck layer. This helps in preserving the representational power while reducing complexity.
- **Efficient Layer Design:** The overall design focuses on making the network smaller and faster, allowing it to be deployed on edge devices while maintaining reasonable accuracy for a wide range of tasks.

Transfer Learning Using MobileNetV2:

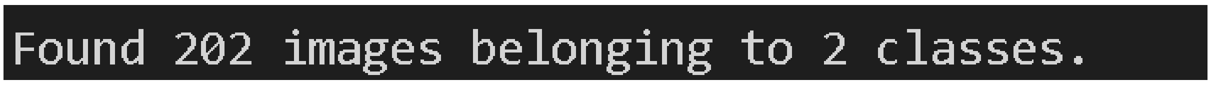
When implementing transfer learning with MobileNetV2 for image classification, the steps typically involve:

1. **Loading the Pre-trained Model:** MobileNetV2, pre-trained on ImageNet (a dataset with over 1 million images across 1,000 classes), is loaded without the final classification layer.
2. **Freezing Base Layers:** The base layers of the model (those trained on ImageNet) are frozen to preserve the learned features (such as edges, textures, shapes).
3. **Adding a Custom Classifier:** A new classification head is added, typically consisting of a fully connected layer (Dense layer) followed by a softmax activation function. This classifier corresponds to the number of categories in the new task.
4. **Training the New Classifier:** Only the added classifier layers are trained on the new dataset, while the pre-trained layers remain frozen.
5. **Fine-Tuning (Optional):** In some cases, after training the classifier, a few of the earlier layers can be unfrozen, and the entire model is trained with a lower learning rate. This fine-tuning helps adapt the model further to the new task, improving performance.

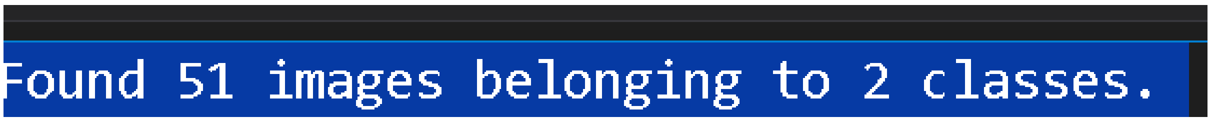
CODE AND OUTPUT-

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

#from keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_data_generator=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)

training_set=train_data_generator.flow_from_directory(r"output\train",target_size=(224,224),batch_size=32,class_mode='binary')

test_data_generator=ImageDataGenerator(rescale=1./255)
```

testing_set=test_data_generator.flow_from_directory(r"output\val",target_size=(224,224),batch_size=32,class_mode='binary')

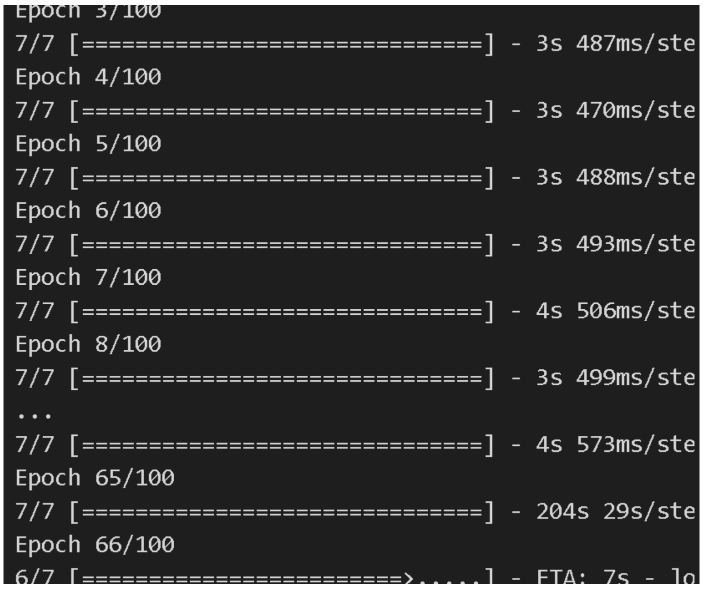


```
Found 202 images belonging to 2 classes.
```



```
ound 51 images belonging to 2 classes.
```

```
from tensorflow.keras.callbacks import EarlyStopping
base_model = tf.keras.applications.MobileNetV2(input_shape=(224,224, 3),
                                               include_top=False,
                                               weights='imagenet')


base_model.trainable = False


model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(10, activation='softmax')  # Adjust the output units to match the
number of classes
])


model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# early_stopping=EarlyStopping(patience=3)


model.fit(training_set, epochs=100, validation_data=testing_set)
        #   ,callbacks=[early_stopping])
```



```
Epoch 3/100
7/7 [==============================] - 3s 487ms/ste
Epoch 4/100
7/7 [==============================] - 3s 470ms/ste
Epoch 5/100
7/7 [==============================] - 3s 488ms/ste
Epoch 6/100
7/7 [==============================] - 3s 493ms/ste
Epoch 7/100
7/7 [==============================] - 4s 506ms/ste
Epoch 8/100
7/7 [==============================] - 3s 499ms/ste
...
7/7 [==============================] - 4s 573ms/ste
Epoch 65/100
7/7 [==============================] - 204s 29s/ste
Epoch 66/100
6/7 [========================>.....] - ETA: 7s - lo
```

VIVA QUESTIONS-

1. **What is MobileNetV2?**
   - MobileNetV2 is a lightweight convolutional neural network designed for mobile and edge devices, utilizing depthwise separable convolutions to reduce complexity and size.
2. **Why do we use MobileNetV2 for transfer learning?**
   - Due to its efficiency and low computational requirements, it is suitable for resource-constrained environments like mobile and embedded devices.
3. **What are depthwise separable convolutions?**
   - Depthwise separable convolutions divide a standard convolution into two simpler operations: depthwise convolution and pointwise convolution, drastically reducing the number of parameters.
4. **What is the importance of freezing layers in transfer learning?**
   - Freezing layers ensures that the pre-trained weights are not modified during initial training, preserving the learned features.
5. **At which point should you unfreeze layers during transfer learning?**
   - After training the new classifier on top of the frozen base model, you may unfreeze some layers and fine-tune the model to improve performance on the new task.