EXPT.NO:10	TRANSFER LEARNING WITH A PRE-TRAINED MODEL
DATE:3/10/2025	

OBJECTIVE

To implement transfer learning using a pre-trained TensorFlow/Keras convolutional neural network (CNN) model by adding custom layers, training on a new dataset, saving and loading the model, and evaluating its performance.

INTRODUCTION

Deep learning models, especially convolutional neural networks (CNNs), require massive amounts of labeled data and computational resources for effective training. Transfer learning addresses this problem by reusing a pre-trained model (trained on a largedataset such as ImageNet) as the starting point for a new but related task.

Instead of training a CNN from scratch, we transfer the learned feature representations and fine-tune them for our custom dataset. This approach significantly reduces training time, improves accuracy, and requires fewer resources.

Methodology

1. Using the Convolutional Base

- A convolutional base refers to thefeature-extraction layers (convolution + pooling) of a trained model such as VGG16, ResNet50, InceptionV3, or MobileNet.
- The convolutional base captures low-level and high-level visual features from input images.
- In transfer learning, we typicallyfreezethese layers to retain the learned weights.

Example:

```
from tensorflow.keras.applications  import VGG16
conv_base= VGG16(weights="imagenet", include_top=False, input_shape=(150, 150, 3))
conv_base.trainable = False
```

2.Adding Custom Layers

On top of the convolutional base, we add custom fully connected layers (denselayers) tailored for our dataset.

Dropout can be used to avoid overfitting.

Example:

```
from tensorflow.kerasimport models, layers
model = models.Sequential([
   conv_base,
   layers.Flatten(),
   layers.Dense(256, activation='relu'),
   layers.Dropout(0.5),
   layers.Dense(1, activation='sigmoid') # for binary classification
])
```

3.Compiling and Training the Model

- The model is compiled with a suitable optimizerand loss function.
- Trainingis done on the custom dataset, usually with data augmentation to improve generalization.

Example:

```
from tensorflow.keras import optimizers
model.compile(optimizer=optimizers.Adam(learning_rate=1e-4),
          loss='binary_crossentropy',
          metrics=['accuracy'])
history= model.fit(train_generator,
              epochs=10,
              validation_data=validation_generator)
```

4.Saving and Loading the Model

After training, themodel can be saved for future usewithout retraining.

Example:

```
# Save
model.save("transfer_learning_model.h5")
# Load
from tensorflow.keras.models import load_model
loaded_model=load_model("transfer_learning_model.h5")
```

5.Evaluating Performance

Model performance is evaluated using test data.

Accuracy, precision, recall, and F1-score are common metrics.

Example:

```
test_loss, test_acc = loaded_model.evaluate(test_generator)
print("Test Accuracy:", test_acc)
```

Real-World Applications of Transfer Learning

1. Medical Imaging – Detecting diseases likepneumonia, cancer, or COVID-19 from X-rays and CT scans.
2. Autonomous Vehicles – Object detection and lane detection for self-drivingcars.
3. Natural Language Processing (NLP) – Using pre-trained models like BERT or GPT for text classification, sentiment analysis, and machine translation.
4. Face Recognition & Security – Identifyingindividuals using facial features.
5. Agriculture– Plant disease detection and crop healthmonitoring using image classification.

Advantages of Using Pre-Trained Models

a. Reduced Training Time:

Since the base network is alreadytrained, only custom layersneed training.

b. Better Accuracy:

Leveragespowerful feature extractionlearned from largedatasets.

c. Less Data Requirement:

Works well even withsmall custom datasets.

d. Computational Efficiency:

Saves resources compared totraining from scratch

Conclusion

Transfer learning with pre-trained TensorFlow/Keras models is a powerful technique that enables high performance with limited dataand computational resources. By reusing a convolutional base and adding custom layers, we can adapt general-purpose models to solve specific real-world problems efficiently.

2303717624321023