Name: Priyanka A

School: Computing and Data Sciences

Email: priyanka.a-26@scds.saiuniversity.edu.in

Dataset

The Car Images Dataset dataset is used for this project.

Original Dataset

1. Total images: 4165
2. Classes: 7 ['Audi', 'Hyundai Creta', 'Mahindra Scorpio', 'Rolls Royce', 'Swift', 'Tata Safari', 'Toyota Innova']
3. Images per class: [1013, 338, 391, 385, 526, 547, 965]
4. Training samples: 3123, Testing samples: 1042

Subset Dataset

To ease training, a selection of 200 samples per category was used to form the subset dataset.

1. **Total images: 1400**
2. **Classes: 7 ['Audi', 'Hyundai Creta', 'Mahindra Scorpio', 'Rolls Royce', 'Swift', 'Tata Safari', 'Toyota Innova']**
3. **Images per class: [200, 200, 200, 200, 200, 200, 200]**
4. **Training samples: 1050, Testing samples: 350**

The subset dataset is split into 75% training and 25% testing.

Pretrained models

Task is to train the following three pretrained CNN models by applying both Transfer-Learning and Fine-Tuning.

1. Model-1: ResNet101V2
2. Model-2: InceptionResNetV2
3. Model-3: DenseNet201

Part 1

Model-3: DenseNet201

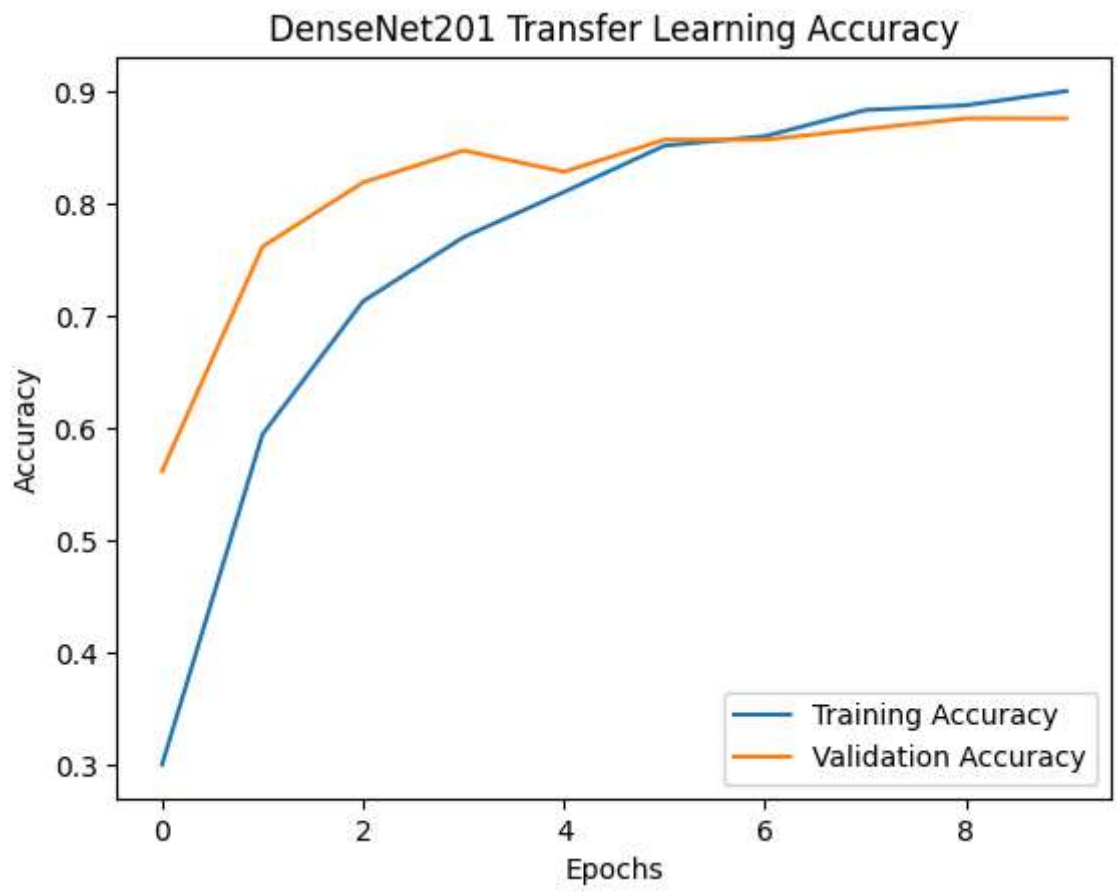
Subtask 1: *Apply the following modifcations to the default classifier layers of the model during Transfer-Learning:*

1. Model-3 → **Include a Dropout layer with 15% drop rate before the output layer.**
2. Train the model for **10 epochs**, and preserve the best performing TL model using the callback. Use 10% of the training dataset for validation.

```
# Import necessary libraries
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
import os
from glob import glob
import matplotlib.pyplot as plt
from PIL import Image

# Set random seeds for reproducibility
np.random.seed(42)
tf.random.set_seed(42)

# Subset the dataset
dataset_path = "../input/car-images-dataset/Car_Dataset"
categories = ["Audi", "Hyundai Creta", "Mahindra Scorpio", "Rolls Royce", "Swift", "Tata Safari", "Toyota Innova"]

# Load images and labels
images, labels = [], []
```

```
    for i, category in enumerate(categories):
        image_files = glob(os.path.join(dataset_path, category, "*.jpg"))
        selected_files = image_files[:200] # Select subset (200) number of samples per category
        for file in selected_files:
            img = Image.open(file).convert("RGB")  # Ensure all images are RGB
            img = img.resize((224, 224))  # Resizing for model input
            images.append(np.array(img))
            labels.append(i)

    # Convert to numpy arrays
    images = np.array(images) / 255.0  # Normalize
    labels = np.array(labels)

    # Split into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.25, stratify=labels, random_state=42)

    # Check dataset details
    print(f"Total images: {len(images)}")
    print(f"Classes: {categories}")
    print(f"Images per class: {[labels.tolist().count(i) for i in range(len(categories))]}")
    print(f"Training samples: {len(X_train)}, Testing samples: {len(X_test)}")

    # Build the model using DenseNet201
    base_model = keras.applications.DenseNet201(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
    base_model.trainable = False  # Freeze base model layers

    # Add classifier layers
    global_avg_pooling = layers.GlobalAveragePooling2D()(base_model.output)
    dropout = layers.Dropout(0.15)(global_avg_pooling)
    output_layer = layers.Dense(len(categories), activation='softmax')(dropout)

    model = keras.models.Model(inputs=base_model.input, outputs=output_layer)

    # Compile the model
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    # Train the model
    callbacks = [
        keras.callbacks.ModelCheckpoint("densenet201_tl_best.keras", save_best_only=True, monitor='val_accuracy'),
        keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=3)
    ]

    history = model.fit(
        X_train, y_train,
        validation_split=0.1,
        epochs=10,
        batch_size=32,
        callbacks=callbacks
    )

    # Plot accuracy
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.title('DenseNet201 Transfer Learning Accuracy')
    plt.show()

    # Evaluate the model on the test set
    model.load_weights("densenet201_tl_best.keras")
    test_loss, test_accuracy = model.evaluate(X_test, y_test)
    print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")

    #model.summary()
```

```
Total images: 1400
Classes: ['Audi', 'Hyundai Creta', 'Mahindra Scorpio', 'Rolls Royce', 'Swift', 'Tata Safari', 'Toyota Innova']
Images per class: [200, 200, 200, 200, 200, 200, 200]
Training samples: 1050, Testing samples: 350
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet201_weights_tf_dim_ordering
74836368/74836368 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Epoch 1/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 85s 2s/step - accuracy: 0.2496 - loss: 1.9729 - val_accuracy: 0.5619 - val_loss: 1.3873
Epoch 2/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 149ms/step - accuracy: 0.5768 - loss: 1.2873 - val_accuracy: 0.7619 - val_loss: 1.0413
Epoch 3/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 136ms/step - accuracy: 0.7163 - loss: 1.0063 - val_accuracy: 0.8190 - val_loss: 0.8782
Epoch 4/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 137ms/step - accuracy: 0.7719 - loss: 0.8366 - val_accuracy: 0.8476 - val_loss: 0.7622
Epoch 5/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.8144 - loss: 0.7118 - val_accuracy: 0.8286 - val_loss: 0.6857
Epoch 6/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 136ms/step - accuracy: 0.8602 - loss: 0.6196 - val_accuracy: 0.8571 - val_loss: 0.6187
Epoch 7/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 87ms/step - accuracy: 0.8690 - loss: 0.5610 - val_accuracy: 0.8571 - val_loss: 0.5822
Epoch 8/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 135ms/step - accuracy: 0.8824 - loss: 0.5041 - val_accuracy: 0.8667 - val_loss: 0.5437
Epoch 9/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 4s 135ms/step - accuracy: 0.8929 - loss: 0.4682 - val_accuracy: 0.8762 - val_loss: 0.5099
Epoch 10/10
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 88ms/step - accuracy: 0.8971 - loss: 0.4419 - val_accuracy: 0.8762 - val_loss: 0.4868
```




```
11/11 ━━━━━━━━━━━━━━━━━━━━ 15s 2s/step - accuracy: 0.8181 - loss: 0.6188
Test Loss: 0.5814439058303833, Test Accuracy: 0.8371428847312927
```

```
model.save("/kaggle/working/densenet201_tl_best.keras")
```