DL-LAB-9-Activities

*Demonstrate **Transfer Learning** with **CNN** for **image classification** in the following **formats** using any of the pre-trained models of your choice (MobileNetV2/VGG16/ResNet50/etc.,) on Keras/TensorFlow.*

1. **As-it-is mode:** Use a **pre-trained/base model** (as it is) for Prediction on new similar test images. (ex: model trained on large datasets can predict class labels for a variety of input test images)
2. **Feature Extractor mode:** Start with a pre-trained model, **remove/exclude the top layer (or classification head)**, freeze the model (make base model without the top layer, non-trainable), Add a new classification head (create a new model on top of the output of the base model using the *Sequential* API), Compile and train the model with custom dataset with few epochs (<=10). (Make sure the **shape** of the custom dataset **matches** with that of the training data used for the pre-trained model)
3. **Fine-tuning mode:** Use a pre-trained model for a new classification task, **Unfreeze a few of the top layers** of a frozen base model and jointly train both the newly-added classifier layers and the last layers of the base model with custom dataset again with fewer epochs (<=10)
4. Compare the performances of all the three models for custom test dataset.

Quick References:
https://www.kaggle.com/code/jeandedieunyandwi/transfer-learning-with-pretrained-convnets
https://www.tensorflow.org/tutorials/images/transfer_learning

****

Sample Code snippet for Exercise - 2:

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Sequential
import numpy as np

IMG_SHAPE = (224, 224, 3) # MobileNetV2 expects 224x224 input images with 3 color channels

# Load MobileNetV2 model, pre-trained on ImageNet, without the top classification layer
base_model = MobileNetV2(input_shape=IMG_SHAPE,
                         include_top=False,
                         weights='imagenet')
```

```
        #freeze the base model
base_model.trainable = False
      # Add the new top classification layer for the custom dataset
model = Sequential([
    base_model,
    GlobalAveragePooling2D(), # Converts the feature maps to a single 1D
vector
    Dense(1, activation='sigmoid') # A new Dense layer for your specific
number of classes (here, 1 for binary classification)
])
#Compile and train the model (assuming binary classification)
model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```
****