

# Week 6 Learning Activities

Student ID: 105000908

Student name: Tran Hung Quoc Tuan

## 1. Concept of 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. Instead of training a model from scratch with random weights, transfer learning leverages pre-trained models (like MobileNetV2, ResNet, or VGG) that have already learned useful features from a large dataset such as ImageNet.

Difference:

- Training from scratch: all model weights are randomly initialized and trained fully on your dataset, requires a large dataset and lots of computation.
- Transfer learning: starts from pre-trained weights, allowing faster convergence and better performance on smaller datasets.

## 2. What is Fine-tuning

Fine-tuning means unfreezing some of the pre-trained layers (usually the top few convolutional layers) and training them alongside new classifier layers with a low learning rate. This allows the model to adapt more closely to the new dataset while retaining useful features learned from the base model.

Why useful: It helps improve accuracy when your dataset is similar but not identical to the pre-training dataset (e.g., new object types but same image styles).

## 3. Freezing the Convolutional Base

Freezing the convolutional base during feature extraction prevents its pre-trained weights from being updated. This is important because:

- It preserves previously learned low-level features (edges, textures, shapes).
- It reduces computation and prevents overfitting, especially when your dataset is small.

Essentially, you're using the base as a fixed feature extractor.

#### 4. Why Use Data Augmentation

Data augmentation artificially expands your training dataset by applying transformations such as rotation, flipping, zooming, or brightness changes. This helps:

- Reduce overfitting by introducing variation.
- Improve model robustness to unseen data.
- Increase dataset size without collecting new data.

#### 5. Take a screenshot of the code snippet where the pre-trained MobileNetV2 model is loaded without the top classification layers

```
# Create the base model from the pre-trained model MobileNet V2
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                                include_top=False,
                                                weights='imagenet')
```

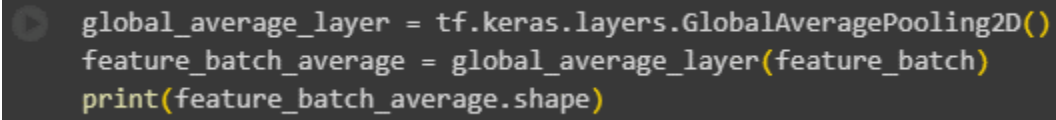
#### 6. Take a screenshot of the portion of code where the pre-trained model is set to be non-trainable for feature extraction purposes.

```
base_model.trainable = False
```

#### 7. Take a screenshot of the data augmentation layers defined in the model.

```
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.RandomFlip('horizontal'),
    tf.keras.layers.RandomRotation(0.2),
])
```

8. Take a screenshot of the code that shows the addition of the new classifier layers on top of the base model.

A screenshot of a Jupyter Notebook cell with a dark background. On the left side of the cell is a circular play button icon. To its right, there are three lines of Python code in a light gray font. The first line defines a global average pooling layer. The second line applies this layer to a feature batch. The third line prints the shape of the resulting feature batch average.

```
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()  
feature_batch_average = global_average_layer(feature_batch)  
print(feature_batch_average.shape)
```