




APPLIED MACHINE LEARNING

# LAB 06

TRAN DUC ANH DANG

103995439



## Explain the concept of transfer learning. How does it differ from training a model from scratch?

Transfer learning is a concept that utilizes a pretrained model on a new, related task. The difference between transfer learning and training a model from scratch is that transfer learning starts with patterns learned from a previous task, while training from scratch involves learning from zero without pre-existing knowledge.

## What is fine-tuning in the context of transfer learning, and why is it useful?

Finetuning in the context of transfer learning is slightly adjusting the deeper layers of a pretrained model to better fit with a specific task. It enhances the model performance on the specific task by adapting pre-learned features to new similar data.

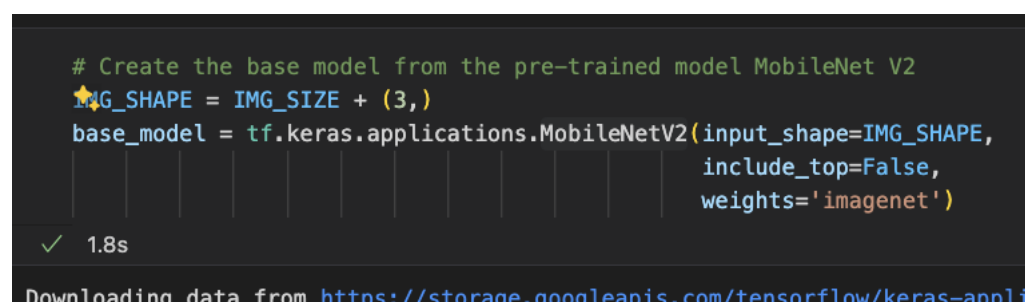
## Why is it important to freeze the convolutional base during feature extraction?

The purpose of freezing convolutional base during feature extraction is to preserve the general features already learned and only adjust specific details for a new task. It is important that it prevents overfitting by keeping the foundational layers stable, which is crucial when training data for the new task is limited.

## Why use data augmentation?

Data augmentation is to increase the diversity of training data through transformations like rotation, scaling, and cropping. It helps the model generalize better, reducing the risk of overfitting, especially when the amount of training data is small.

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

A screenshot of a code editor showing the loading of a MobileNetV2 model. The code defines an input shape and creates the model with 'include\_top=False' and 'weights='imagenet''. A green checkmark and '1.8s' indicate successful execution. Below the code, a message shows data being downloaded from a Google Cloud Storage link.

```
# 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')
```

✓ 1.8s

Downloading data from <https://storage.googleapis.com/tensorflow/keras-applications>

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

A screenshot of a code editor showing the line 'base\_model.trainable = False'. A green checkmark and '0.0s' indicate successful execution.

```
base_model.trainable = False
```

✓ 0.0s

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

```
> ✓ 0.0s  
data_augmentation = tf.keras.Sequential([  
    tf.keras.layers.RandomFlip('horizontal'),  
    tf.keras.layers.RandomRotation(0.2),  
])
```

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

```
✓ 0.0s  
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()  
feature_batch_average = global_average_layer(feature_batch)  
print(feature_batch_average.shape)  
(32, 1280)  
  
Apply a tf.keras.layers.Dense layer to convert these features into a single value.  
  
✓ 0.0s  
prediction_layer = tf.keras.layers.Dense(1, activation='sigmoid')  
prediction_batch = prediction_layer(feature_batch_average)  
print(prediction_batch.shape)  
(32, 1)  
  
Build a model by chaining together the data augmentation, rescaling, base_model, and the new classifier layers.  
  
✓ 0.0s  
inputs = tf.keras.Input(shape=(160, 160, 3))  
x = data_augmentation(inputs)  
x = preprocess_input(x)  
x = base_model(x, training=False)  
x = global_average_layer(x)  
x = tf.keras.layers.Dropout(0.2)(x)  
outputs = prediction_layer(x)  
model = tf.keras.Model(inputs, outputs)
```