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

Transfer learning is about re-using a pre-trained model for a different task or other purposes.

Training from scratch starts with random weight, learning both general and specific features. On the other hand, transfer learning train using pre-learned features emphasize on task-specific learning

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

Fine tuning in this context involves taking a pre-trained model and adjusting its weights somehow on a new dataset

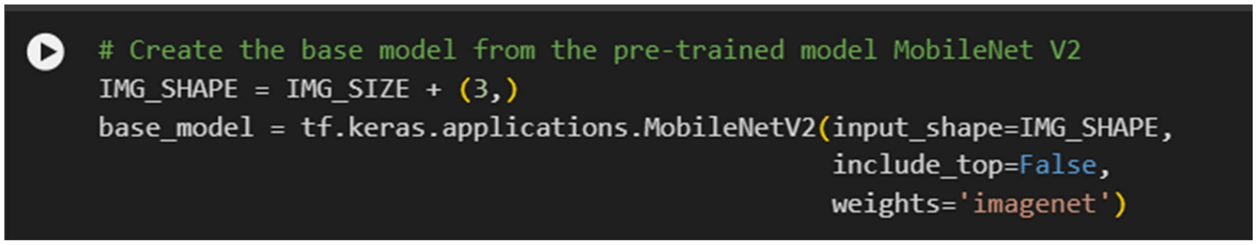
Why is it important to freeze the convolutional base during feature extraction?

Because we need to preserve the already learned general features and only re-train specific details for new task. Another point is that if the training data for the new task is low, training too many layers of the pre-train model can lead to severe overfitting

Why use data augmentation?

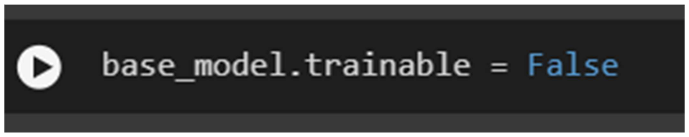
we use data augmentation to increase the diversity of the training data especially when the training data is lacking. This help the model to generalize better and hopefully narrowing the gap of overfitting

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



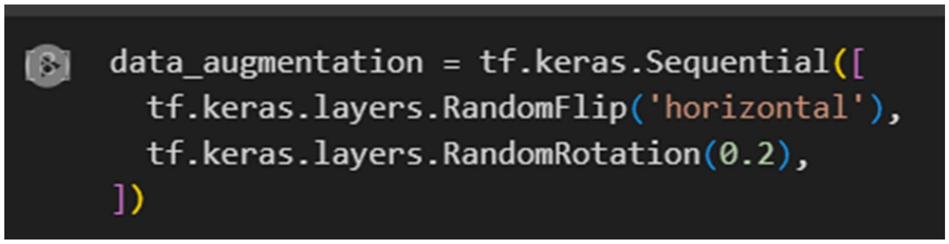
```
# Create the base model from the pre-trained model MobileNet V2
IMG_SHAPE = IMG_SIZE + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

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



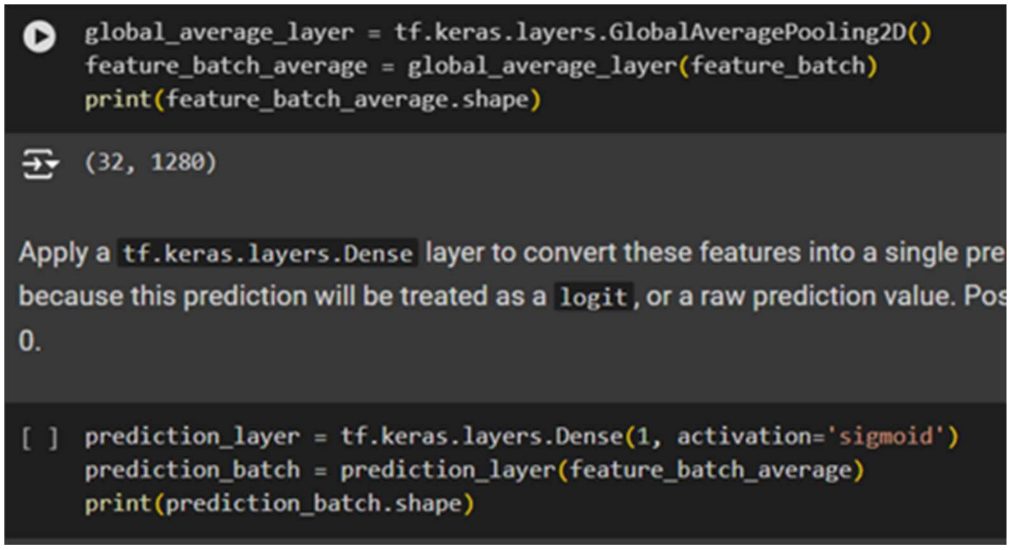
```
base_model.trainable = False
```

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),
])
```

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



```
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 pre
because this prediction will be treated as a `logit`, or a raw prediction value. Pos
0.

```
[ ] prediction_layer = tf.keras.layers.Dense(1, activation='sigmoid')
    prediction_batch = prediction_layer(feature_batch_average)
    print(prediction_batch.shape)
```