base_model_finetuning

April 11, 2025

[ ]:
```
import tensorflow as tf
import tensorflow_datasets as tfds
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
```

[ ]:
```
!wget https://raw.githubusercontent.com/mrdbourke/tensorflow-deep-learning/main/
 ↪extras/helper_functions.py
from helper_functions import plot_loss_curves, compare_historys
```

```
--2025-02-15 08:02:21--  https://raw.githubusercontent.com/mrdbourke/tensorflow-
deep-learning/main/extras/helper_functions.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 10246 (10K) [text/plain]
Saving to: ‘helper_functions.py’

 helper_functions.py   0%[                    ]       0  --.-KB/s
helper_functions.py 100%[===================>]  10.01K  --.-KB/s    in 0.001s

2025-02-15 08:02:21 (18.8 MB/s) - ‘helper_functions.py’ saved [10246/10246]
```

[ ]:
```
# Set hyperparameters
BATCH_SIZE = 64
IMG_SIZE = 96  # Upscale CIFAR-10 images (32x32) to a larger size for␣
 ↪MobileNetV2
AUTOTUNE = tf.data.AUTOTUNE
```

[ ]:
```
(ds_train, ds_test), ds_info = tfds.load(
    'cifar10',
    split=['train', 'test'],
    as_supervised=True,
    with_info=True
)
```

```
Downloading and preparing dataset 162.17 MiB (download: 162.17 MiB, generated:
132.40 MiB, total: 294.58 MiB) to /root/tensorflow_datasets/cifar10/3.0.2…

Dl Completed…: 0 url [00:00, ? url/s]

Dl Size…: 0 MiB [00:00, ? MiB/s]

Extraction completed…: 0 file [00:00, ? file/s]

Generating splits…:   0%|          | 0/2 [00:00<?, ? splits/s]

Generating train examples…:   0%|          | 0/50000 [00:00<?, ? examples/s]

Shuffling /root/tensorflow_datasets/cifar10/incomplete.ICZA7D_3.0.2/
  ↪cifar10-train.tfrecord*…:   0%|          …

Generating test examples…:   0%|          | 0/10000 [00:00<?, ? examples/s]

Shuffling /root/tensorflow_datasets/cifar10/incomplete.ICZA7D_3.0.2/cifar10-test.
  ↪tfrecord*…:   0%|          …

Dataset cifar10 downloaded and prepared to
/root/tensorflow_datasets/cifar10/3.0.2. Subsequent calls will reuse this data.
```

[ ]:
```
ds_info.features
```

[ ]:
```
FeaturesDict({
    'id': Text(shape=(), dtype=string),
    'image': Image(shape=(32, 32, 3), dtype=uint8),
    'label': ClassLabel(shape=(), dtype=int64, num_classes=10),
})
```

[ ]:
```
class_names = ds_info.features['label'].names
class_names
```

[ ]:
```
['airplane',
 'automobile',
 'bird',
 'cat',
 'deer',
 'dog',
 'frog',
 'horse',
 'ship',
 'truck']
```

[ ]:
```
train_sample = ds_train.take(1) # takes one sample from train data
train_sample
```

[ ]:
```
<_TakeDataset element_spec=(TensorSpec(shape=(32, 32, 3), dtype=tf.uint8,
name=None), TensorSpec(shape=(), dtype=tf.int64, name=None))>
```

[ ]:
```
for image, label in train_sample:
  print(f'''
  image_shape : {image.shape}
  image_datatype : {image.dtype}
  target_class : {label}
  class_name : {class_names[label.numpy()]}
  '''
  )
```

```
image_shape : (32, 32, 3)
image_datatype : <dtype: 'uint8'>
target_class : 7
class_name : horse
```

[ ]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(1.5, 1.5))
plt.imshow(image)
plt.title(class_names[label.numpy()])
plt.axis(False)
```

[ ]: (-0.5, 31.5, 31.5, -0.5)

[ ]:
```
len(ds_train), len(ds_test)
```

[ ]: (50000, 10000)

[ ]:
```
#  data augmentation pipeline for training data
data_augmentation = tf.keras.Sequential([
    # Resize to desired IMG_SIZE
    tf.keras.layers.Resizing(IMG_SIZE, IMG_SIZE),
    tf.keras.layers.RandomFlip("horizontal"),
    tf.keras.layers.RandomRotation(0.1),
])
```

[ ]:
```
# Preprocessing function using MobileNetV2's preprocess_input
def preprocess(image, label):
    image = tf.cast(image, tf.float32)
    image = preprocess_input(image)
    return image, label
```

[ ]:
```
# Augmentation function (applied only on training set)
def augment(image, label):
    image = data_augmentation(image)
    return image, label
```

[ ]:
```
# Prepare the training dataset
ds_train = ds_train.map(augment, num_parallel_calls=AUTOTUNE)
ds_train = ds_train.map(preprocess, num_parallel_calls=AUTOTUNE)
ds_train = ds_train.shuffle(1000).batch(BATCH_SIZE).prefetch(AUTOTUNE)
```

[ ]:
```
# For the test dataset, only resize and preprocess (no augmentation)
def resize_and_preprocess(image, label):
    image = tf.cast(image, tf.float32)
    image = tf.image.resize(image, [IMG_SIZE, IMG_SIZE])
    image = preprocess_input(image)
    return image, label
```

[ ]:
```
ds_test = ds_test.map(resize_and_preprocess, num_parallel_calls=AUTOTUNE)
ds_test = ds_test.batch(BATCH_SIZE).prefetch(AUTOTUNE)
```

[ ]:
```
# Build the model using MobileNetV2 as the base
base_model = MobileNetV2(
    include_top=False,
    weights='imagenet',
    input_shape=(IMG_SIZE, IMG_SIZE, 3)
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
ns/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_96_no_top.h5
9406464/9406464             0s
0us/step
```

[ ]:
```
base_model.trainable = False
```

[ ]:
```
inputs = tf.keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = base_model(inputs, training=False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.2)(x)
outputs = tf.keras.layers.Dense(10, activation='softmax')(x)
model = tf.keras.Model(inputs, outputs)
```

[ ]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

[ ]:
```
model.summary()
```

```
Model: "functional_1"
```

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 96, 96, 3)	0
mobilenetv2_1.00_96 (Functional)	(None, 3, 3, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dropout (Dropout)	(None, 1280)	0
dense (Dense)	(None, 10)	12,810

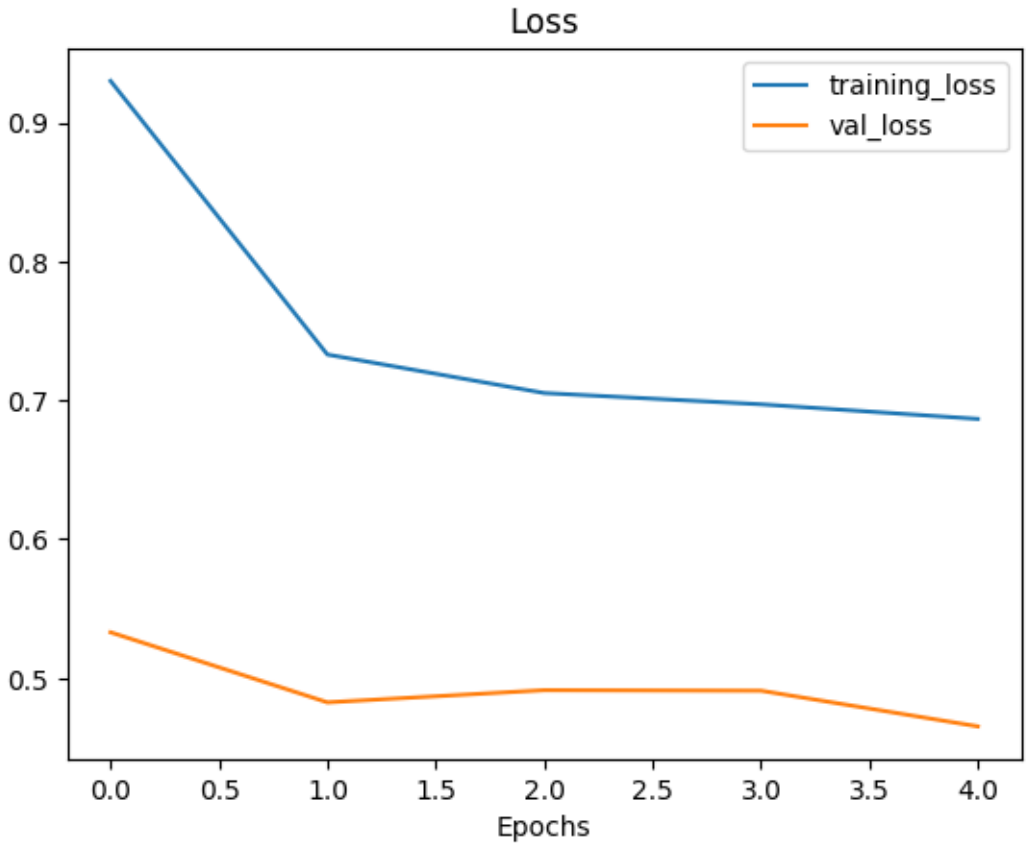
```
Total params: 2,270,794 (8.66 MB)

Trainable params: 12,810 (50.04 KB)

Non-trainable params: 2,257,984 (8.61 MB)
```

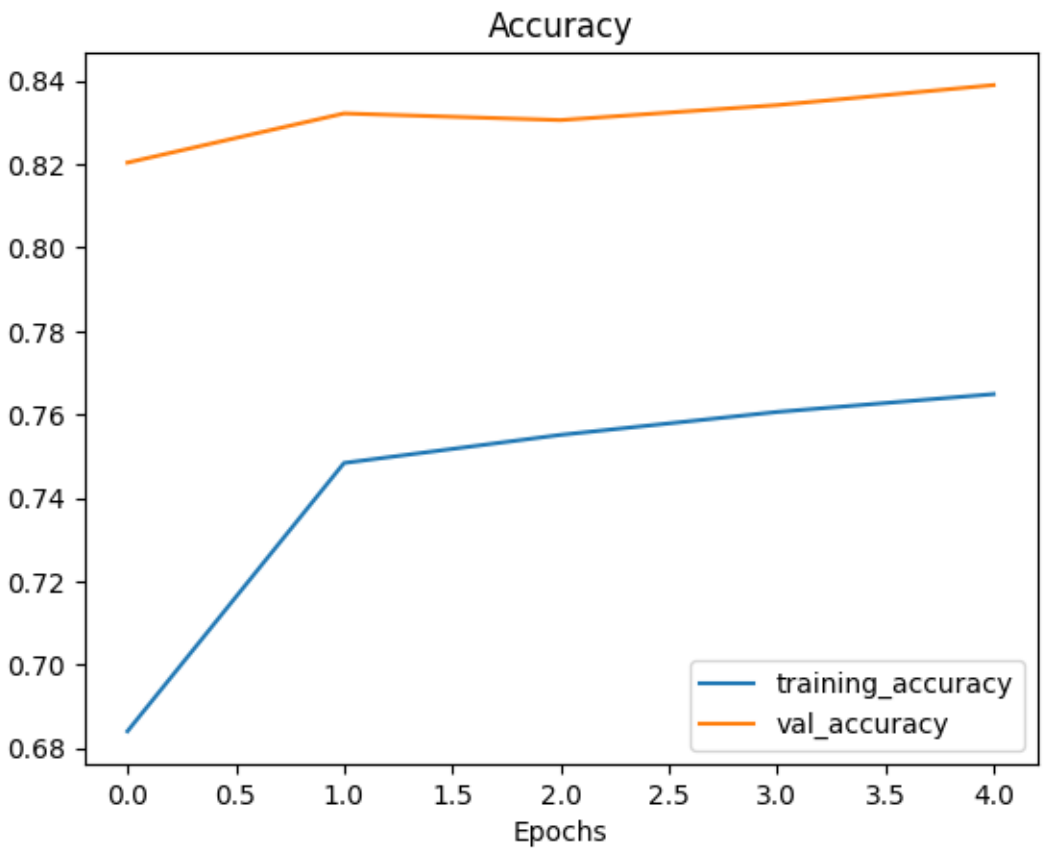
[ ]:
```
history = model.fit(
    ds_train,
    epochs=5,
    validation_data=ds_test
)
```

```
Epoch 1/5
782/782             129s 149ms/step -
```

```
accuracy: 0.5983 - loss: 1.1976 - val_accuracy: 0.8204 - val_loss: 0.5330
Epoch 2/5
782/782             105s 132ms/step -
accuracy: 0.7455 - loss: 0.7420 - val_accuracy: 0.8322 - val_loss: 0.4826
Epoch 3/5
782/782             106s 133ms/step -
accuracy: 0.7520 - loss: 0.7080 - val_accuracy: 0.8306 - val_loss: 0.4913
Epoch 4/5
782/782             104s 131ms/step -
accuracy: 0.7584 - loss: 0.7036 - val_accuracy: 0.8342 - val_loss: 0.4910
Epoch 5/5
782/782             147s 138ms/step -
accuracy: 0.7661 - loss: 0.6819 - val_accuracy: 0.8390 - val_loss: 0.4653
```

[ ]:
```
plot_loss_curves(history)
```





[ ]:
```
base_model.trainable = True
fine_tune_at = 25
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False
```

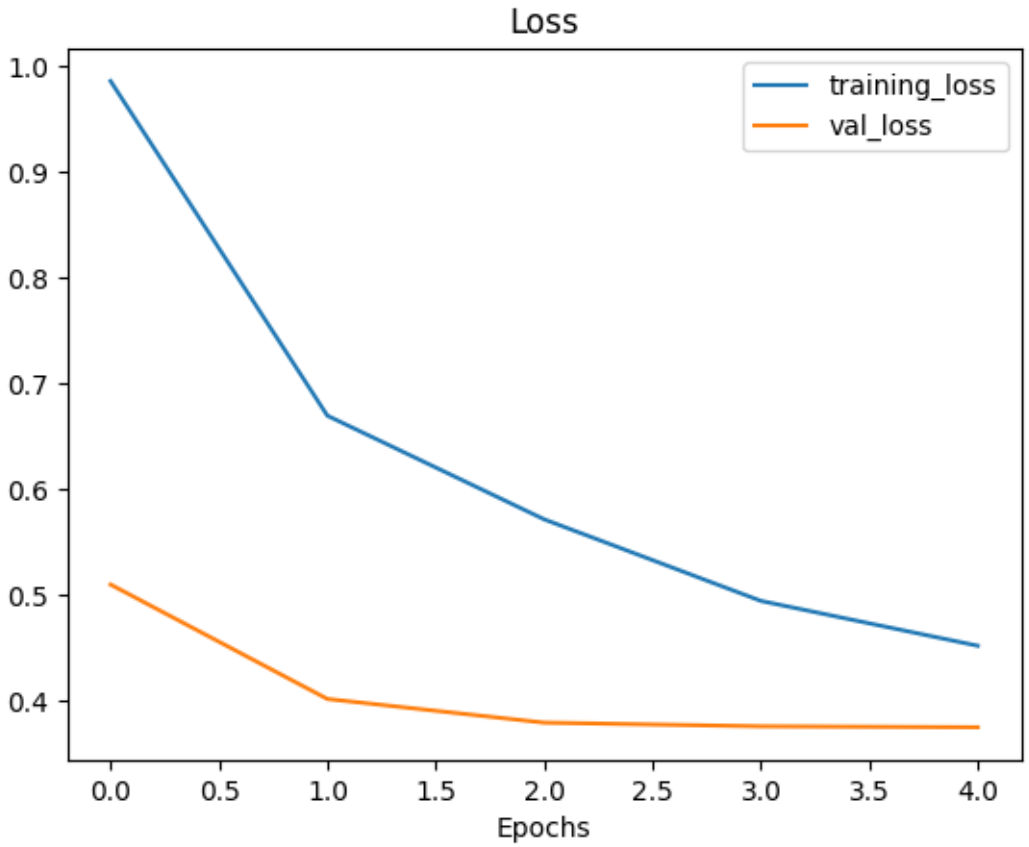
[ ]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(1e-5),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

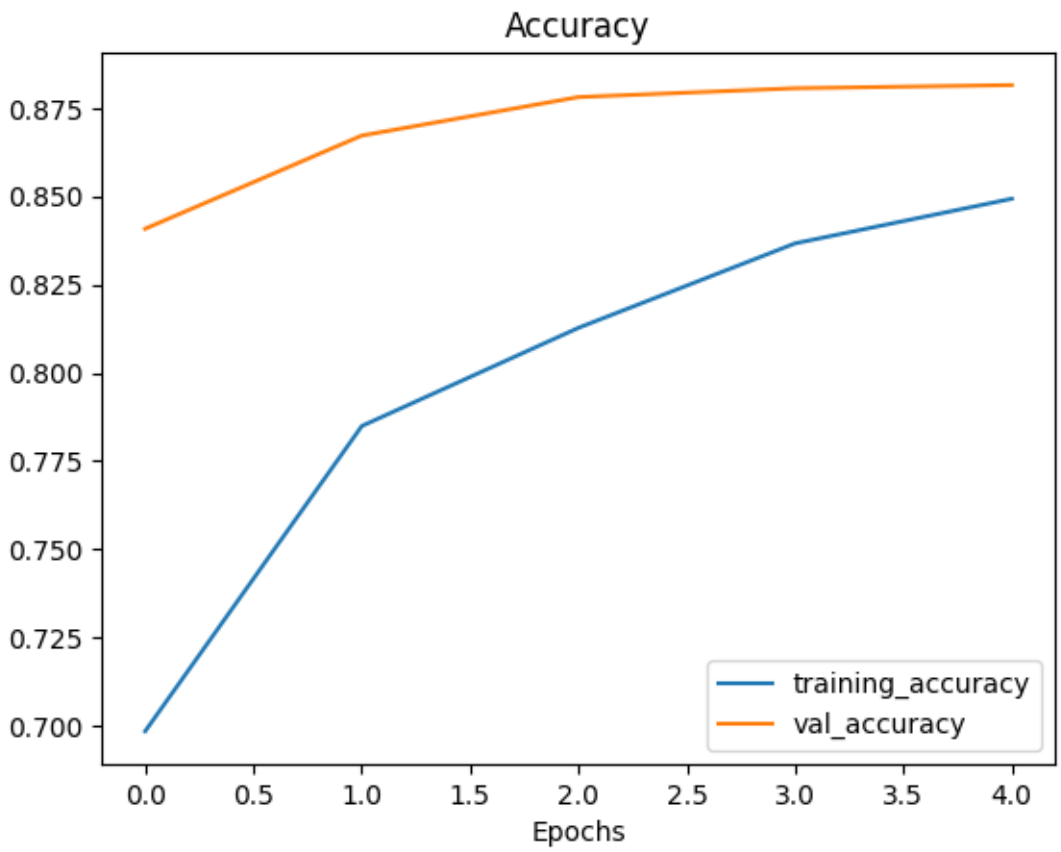
[ ]:
```
history = model.fit(
    ds_train,
    epochs=5,
    validation_data=ds_test,
)
```

```
Epoch 1/5
782/782              206s 197ms/step -
accuracy: 0.6337 - loss: 1.2428 - val_accuracy: 0.8408 - val_loss: 0.5099
Epoch 2/5
782/782              129s 163ms/step -
```

```
accuracy: 0.7762 - loss: 0.7042 - val_accuracy: 0.8672 - val_loss: 0.4018
Epoch 3/5
782/782             139s 160ms/step -
accuracy: 0.8082 - loss: 0.5913 - val_accuracy: 0.8781 - val_loss: 0.3795
Epoch 4/5
782/782             128s 162ms/step -
accuracy: 0.8330 - loss: 0.5092 - val_accuracy: 0.8806 - val_loss: 0.3760
Epoch 5/5
782/782             128s 161ms/step -
accuracy: 0.8473 - loss: 0.4601 - val_accuracy: 0.8815 - val_loss: 0.3751
```

[ ]:
```
plot_loss_curves(history)
```

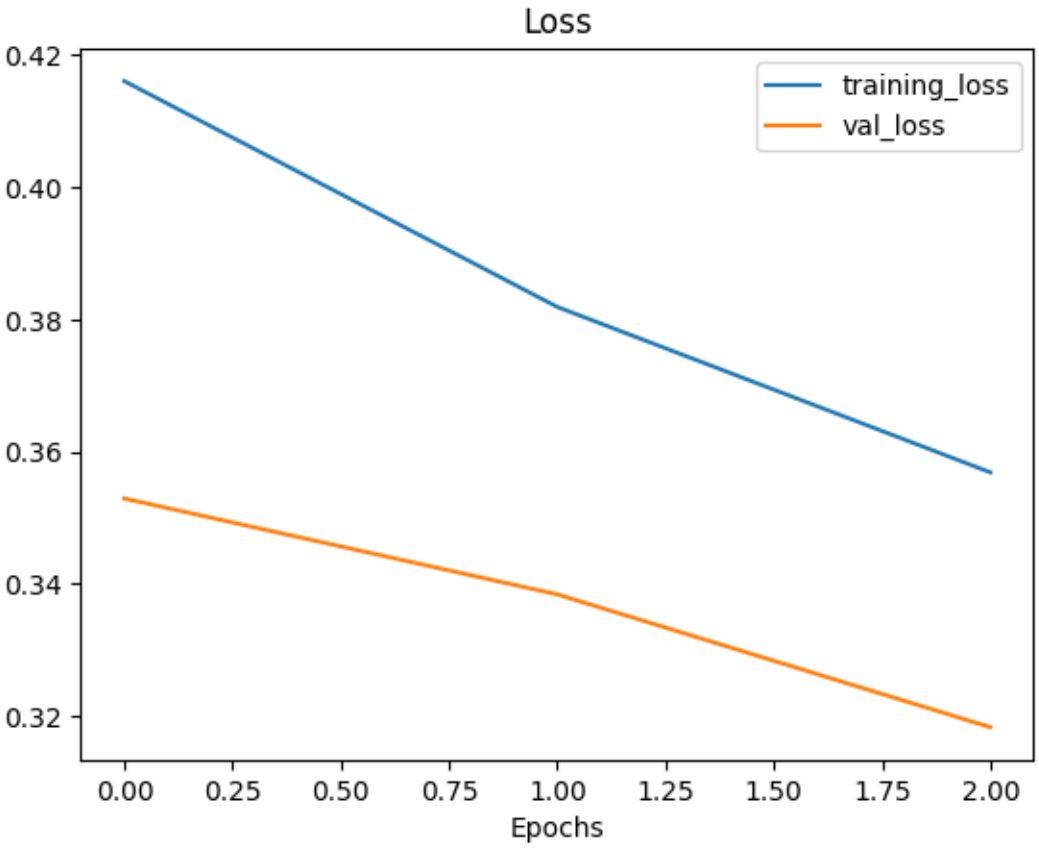


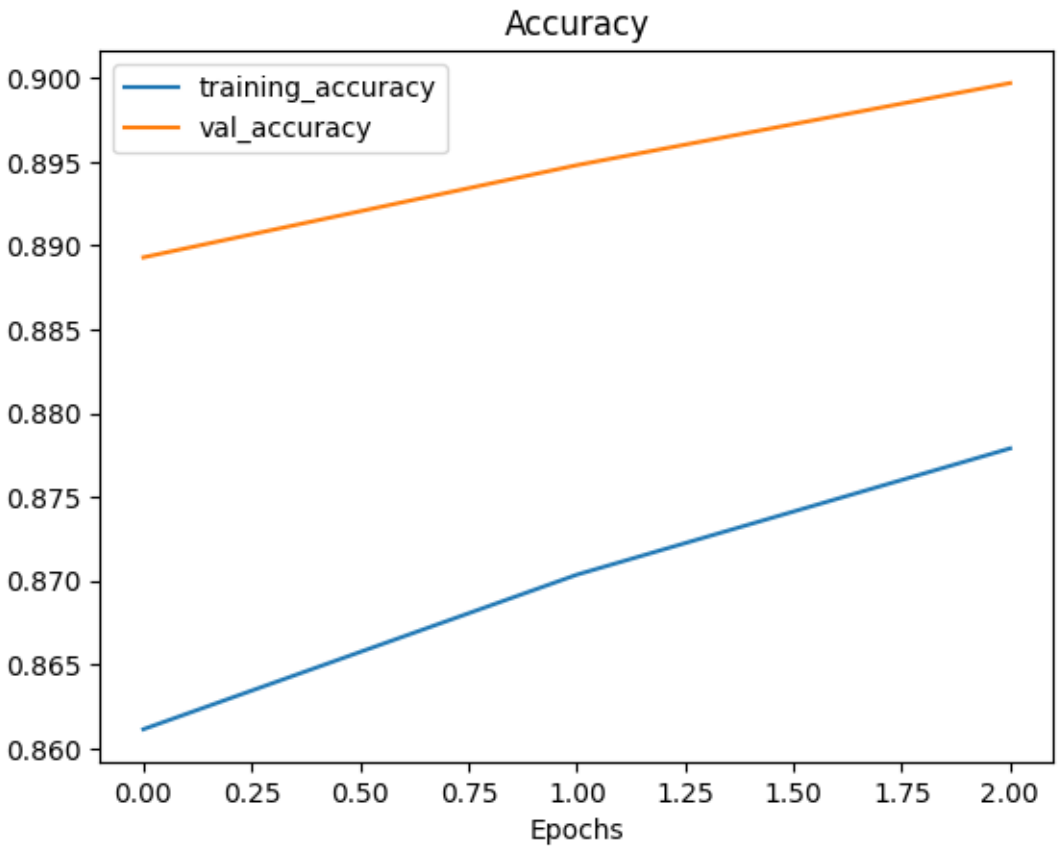


[ ]:
```
history = model.fit(
    ds_train,
    epochs=3,
    validation_data=ds_test,
)
```

```
Epoch 1/3
782/782             130s 164ms/step -
accuracy: 0.8579 - loss: 0.4294 - val_accuracy: 0.8893 - val_loss: 0.3529
Epoch 2/3
782/782             142s 165ms/step -
accuracy: 0.8679 - loss: 0.3882 - val_accuracy: 0.8948 - val_loss: 0.3384
Epoch 3/3
782/782             131s 166ms/step -
accuracy: 0.8745 - loss: 0.3712 - val_accuracy: 0.8997 - val_loss: 0.3182
```

[ ]:
```
plot_loss_curves(history)
```


Loss
training_loss
val_loss
0.42
0.40
0.38
0.36
0.34
0.32
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Epochs

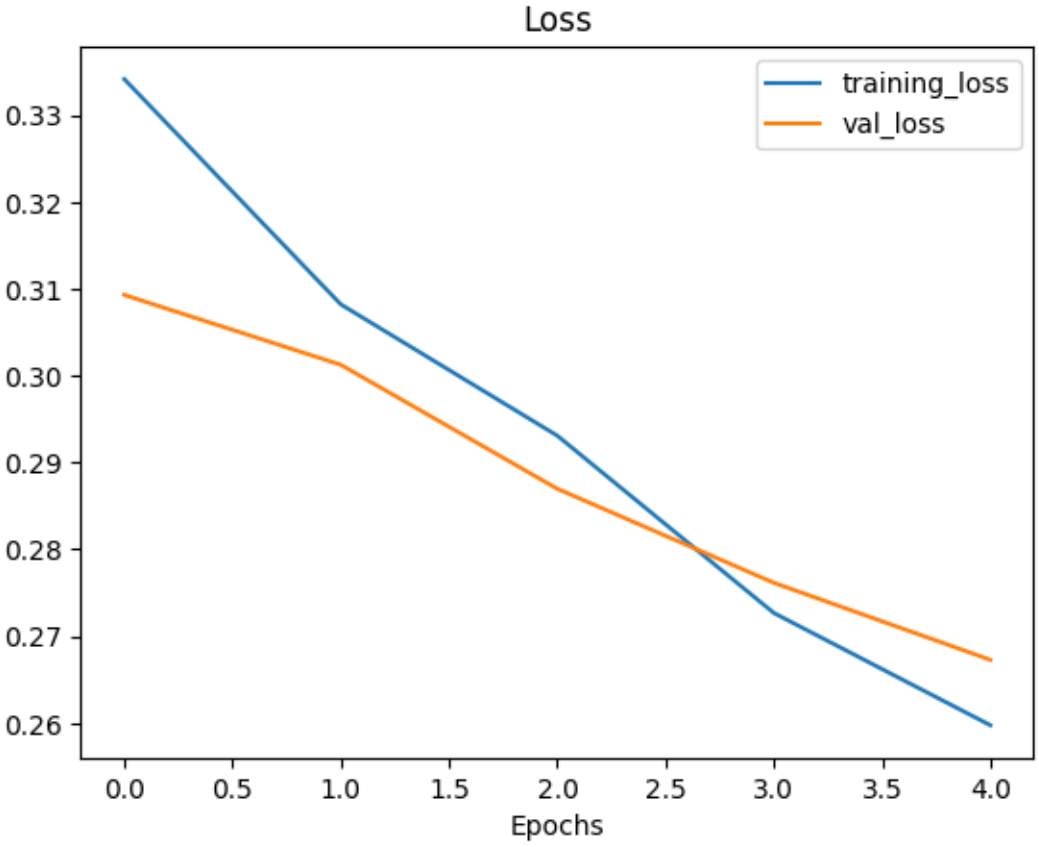


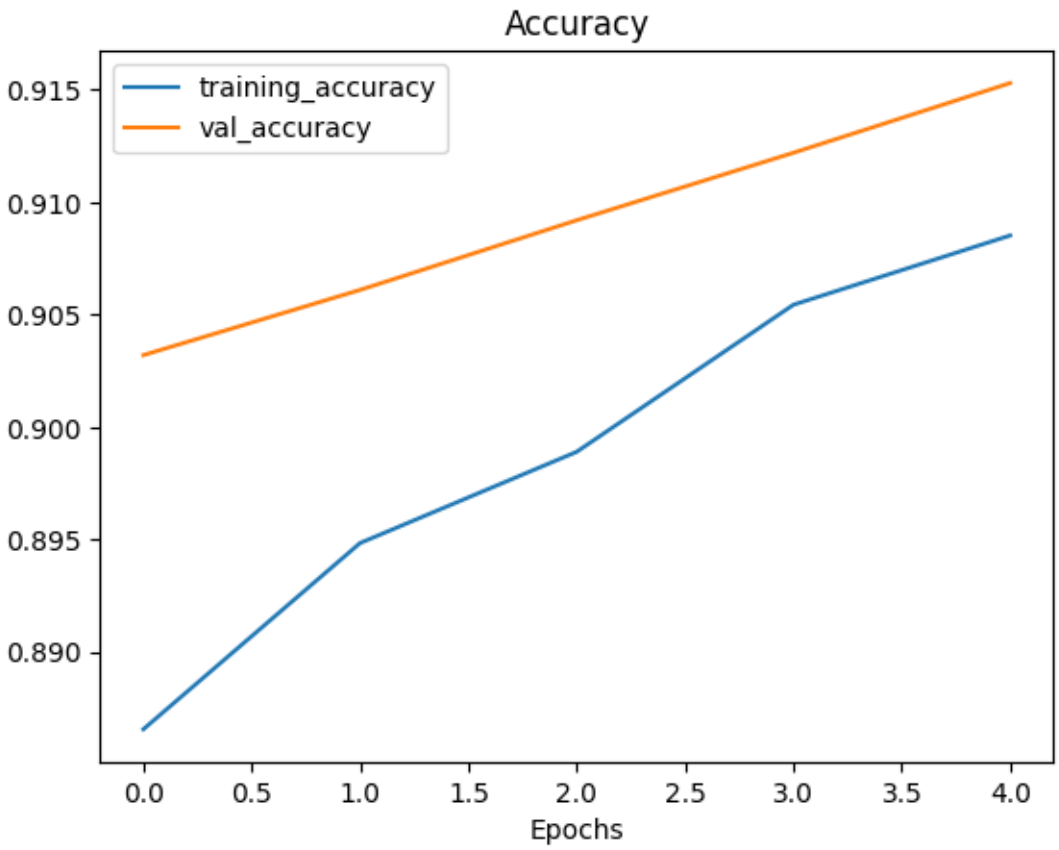
[ ]:
```
history = model.fit(
    ds_train,
    epochs=5,
    validation_data=ds_test,
)
```

```
Epoch 1/5
782/782             133s 166ms/step -
accuracy: 0.8846 - loss: 0.3405 - val_accuracy: 0.9032 - val_loss: 0.3093
Epoch 2/5
782/782             129s 163ms/step -
accuracy: 0.8957 - loss: 0.3091 - val_accuracy: 0.9061 - val_loss: 0.3013
Epoch 3/5
782/782             128s 162ms/step -
accuracy: 0.8972 - loss: 0.2990 - val_accuracy: 0.9092 - val_loss: 0.2870
Epoch 4/5
782/782             128s 161ms/step -
accuracy: 0.9057 - loss: 0.2721 - val_accuracy: 0.9122 - val_loss: 0.2762
Epoch 5/5
782/782             125s 158ms/step -
```

```
accuracy: 0.9049 - loss: 0.2712 - val_accuracy: 0.9153 - val_loss: 0.2673
```

[ ]:
```
plot_loss_curves(history)
```





[ ]:
```
model.save("fine_tuned_model.keras")
model.save("fine_tuned_model.h5", save_format='h5')
```

```
WARNING:absl:The `save_format` argument is deprecated in Keras 3. We recommend
removing this argument as it can be inferred from the file path. Received:
save_format=h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
```

[ ]:
```
loaded_model = tf.keras.models.load_model("fine_tuned_model.keras")
```

[ ]:
```
IMG_SIZE = 96  #matches the training size
BATCH_SIZE = 64
AUTOTUNE = tf.data.AUTOTUNE
```

[ ]:
```
# Load full CIFAR-10 dataset
ds_full, ds_info = tfds.load(
```

```
    'cifar10',
    split='train+test',
    as_supervised=True,
    with_info=True
)
```

[ ]:
```
def preprocess(image, label):
    image = tf.cast(image, tf.float32)
    image = tf.image.resize(image, [IMG_SIZE, IMG_SIZE])
    image = preprocess_input(image)
    return image, label
```

[ ]:
```
# Prepare dataset (apply preprocessing, batch, and prefetch)
ds_full = ds_full.map(preprocess, num_parallel_calls=AUTOTUNE)
ds_full = ds_full.batch(BATCH_SIZE).prefetch(AUTOTUNE)
```

[ ]:
```
loss, accuracy = model.evaluate(ds_full)
loss, accuracy
```

```
938/938            27s 28ms/step -
accuracy: 0.9364 - loss: 0.1922
```

[ ]:
```
(0.16968880593776703, 0.9430000185966492)
```

[ ]: