

```
import kagglehub

path = kagglehub.dataset_download("kaustubhb999/tomatoleaf")

print("Path to dataset files:", path)
```

Downloading from [https://www.kaggle.com/api/v1/datasets/download/kaustubhb999/tomatoleaf?dataset\\_version\\_number=1...](https://www.kaggle.com/api/v1/datasets/download/kaustubhb999/tomatoleaf?dataset_version_number=1...)  
100%|██████████| 179M/179M [00:08<00:00, 21.1MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/kaustubhb999/tomatoleaf/versions/1

```
import os

# Path from KaggleHub download
base_download_path = path
print("Kaggle dataset base path:", base_download_path)

print("\nFolders found:")
for p in os.listdir(base_download_path):
    print(" -", p)

dataset_root = base_download_path
print("\nDetected dataset root:", dataset_root)

class_folders = [f for f in os.listdir(dataset_root) if os.path.isdir(os.path.join(dataset_root, f))]
print("\nDetected class folders:")
for c in class_folders:
    print(" -", c)
```

Kaggle dataset base path: /root/.cache/kagglehub/datasets/kaustubhb999/tomatoleaf/versions/1

Folders found:  
- tomato

Detected dataset root: /root/.cache/kagglehub/datasets/kaustubhb999/tomatoleaf/versions/1

Detected class folders:  
- tomato

```
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import os

TRAIN_DIR = dataset_root
VAL_DIR = None
dataset_mode = "split_from_train"

IMG_SIZE = 224
BATCH_SIZE = 32

print("Using TRAIN_DIR:", TRAIN_DIR)
print("No validation folder found. Splitting 20% of TRAIN_DIR for validation.")

def build_from_dirs(train_dir, val_dir=None, img_size=IMG_SIZE, batch_size=BATCH_SIZE):

    seed = 123
    train_ds = tf.keras.utils.image_dataset_from_directory(
        train_dir,
        image_size=(img_size, img_size),
        batch_size=batch_size,
        label_mode='categorical',
        validation_split=0.2,
        subset='training',
        seed=seed,
        shuffle=True
    )

    val_ds = tf.keras.utils.image_dataset_from_directory(
        train_dir,
        image_size=(img_size, img_size),
        batch_size=batch_size,
        label_mode='categorical',
        validation_split=0.2,
        subset='validation',
        seed=seed,
        shuffle=False
    )

    AUTOTUNE = tf.data.AUTOTUNE
```

```

train_ds = train_ds.map(lambda x, y: (preprocess_input(x), y), num_parallel_calls=AUTOTUNE)
val_ds   = val_ds.map(lambda x, y: (preprocess_input(x), y), num_parallel_calls=AUTOTUNE)

train_ds = train_ds.cache().shuffle(1000).prefetch(AUTOTUNE)
val_ds   = val_ds.cache().prefetch(AUTOTUNE)

return train_ds, val_ds

train_ds, val_ds = build_from_dirs(TRAIN_DIR, VAL_DIR)

for _, y in train_ds.take(1):
    num_classes = y.shape[-1]

print("Number of classes detected:", num_classes)

```

Using TRAIN\_DIR: /root/.cache/kagglehub/datasets/kaustubhb999/tomatoleaf/versions/1  
 No validation folder found. Splitting 20% of TRAIN\_DIR for validation.  
 Found 11000 files belonging to 1 classes.  
 Using 8800 files for training.  
 Found 11000 files belonging to 1 classes.  
 Using 2200 files for validation.  
 Number of classes detected: 1

```

from tensorflow.keras import layers

data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.06),
    layers.RandomZoom(0.05),
], name="data_augmentation")

```

```

from tensorflow.keras import models, layers
from tensorflow.keras.applications import MobileNetV2

base_model = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(IMG_SIZE, IMG_SIZE, 3)
)

base_model.trainable = False # Freeze for transfer learning

inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))

# Augmentation
x = data_augmentation(inputs)

# Base model
x = base_model(x, training=False)

# Pooling
x = layers.GlobalAveragePooling2D()(x)

# Dense Head
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.3)(x)

x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.2)(x)

# Output layer
outputs = layers.Dense(num_classes, activation='softmax')(x)

model = models.Model(inputs, outputs)

```

Downloading data from [https://storage.googleapis.com/tensorflow/keras-applications/mobilenet\\_v2/mobilenet\\_v2\\_weights\\_tf\\_dim\\_9406464/9406464](https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_9406464/9406464) 2s 0us/step

```

model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

EPOCHS = 10

from tensorflow.keras import callbacks

```

```
callbacks_list = [
    callbacks.ModelCheckpoint('best_model.h5', monitor='val_accuracy', save_best_only=True),
    callbacks.EarlyStopping(monitor='val_accuracy', patience=3, restore_best_weights=True),
    callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2)
]
```

```
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    callbacks=callbacks_list,
    verbose=1
)
```

```
Epoch 1/10
275/275 ————— 492s 2s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 2/10
275/275 ————— 507s 2s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 3/10
275/275 ————— 497s 2s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/10
275/275 ————— 494s 2s/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
```

```
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

def build_test_ds(train_dir, img_size=IMG_SIZE, batch_size=BATCH_SIZE):
    seed = 123
    test_ds = tf.keras.utils.image_dataset_from_directory(
        train_dir,
        image_size=(img_size, img_size),
        batch_size=batch_size,
        label_mode='categorical',
        validation_split=0.1,
        subset='validation',
        seed=seed,
        shuffle=False
    )

    AUTOTUNE = tf.data.AUTOTUNE
    test_ds = test_ds.map(lambda x, y: (preprocess_input(x), y), num_parallel_calls=AUTOTUNE)
    test_ds = test_ds.cache().prefetch(AUTOTUNE)
    return test_ds

test_ds = build_test_ds(TRAIN_DIR)

num_test_examples = sum(1 for _ in test_ds.unbatch())
print(f"Test dataset ready - total test images: {num_test_examples}")

loss, acc = model.evaluate(test_ds)
print(f"Test Accuracy: {acc*100:.2f}% - Test Loss: {loss:.4f}")
```

```
Found 11000 files belonging to 1 classes.
Using 1100 files for validation.
Test dataset ready - total test images: 1100
35/35 ————— 40s 1s/step - accuracy: 1.0000 - loss: 0.0000e+00
Test Accuracy: 100.00% - Test Loss: 0.0000
```

