


```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("tushar5sharma/plant-village-dataset-updated")

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

 Path to dataset files: /kaggle/input/plant-village-dataset-updated

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import pandas as pd
import os
from PIL import Image
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.metrics import Precision, Recall
from tensorflow.keras import regularizers
from tensorflow.keras.callbacks import ModelCheckpoint
import seaborn as sns
import matplotlib.pyplot as plt
```

path = "/kaggle/input/plant-village-dataset-updated"


+ Code

+ Text

```
classes = [
    "Apple Healthy", "Apple Apple Scab", "Apple Black Rot", "Apple Cedar Apple Rust",
    "Bell Pepper Healthy", "Bell Pepper Bacterial Spot",
    "Cherry Healthy", "Cherry Powdery Mildew",
    "Corn (Maize) Healthy", "Corn (Maize) Cercospora Leaf Spot",
    "Corn (Maize) Common Rust ", "Corn (Maize) Northern Leaf Blight",
    "Grape Healthy", "Grape Black Rot", "Grape Esca (Black Measles)", "Grape Leaf Blight",
    "Peach Healthy", "Peach Bacterial Spot",
    "Potato Healthy", "Potato Early Blight", "Potato Late Blight",
    "Strawberry Healthy", "Strawberry Leaf Scorch",
    "Tomato Healthy", "Tomato Bacterial Spot", "Tomato Early Blight",
    "Tomato Late Blight", "Tomato Septoria Leaf Spot", "Tomato Yellow Leaf Curl Virus"
]
```

```
name_to_index = {name: index for index, name in enumerate(classes)}
index_to_name = {index: name for index, name in enumerate(classes)}
```

```
print(name_to_index)
print(index_to_name)
```

 { 'Apple Healthy': 0, 'Apple Apple Scab': 1, 'Apple Black Rot': 2, 'Apple Cedar Apple Rust': 3, 'Bell Pepper Healthy': 4, 'Bell Pepper Bacterial Spot': 5, 'Cherry Healthy': 6, 'Cherry Powdery Mildew': 7, 'Corn (Maize) Healthy': 8, 'Corn (Maize) Cercospora Leaf Spot': 9, 'Corn (Maize) Common Rust': 10, 'Corn (Maize) Northern Leaf Blight': 11, 'Grape Healthy': 12, 'Grape Black Rot': 13, 'Grape Esca (Black Measles)': 14, 'Grape Leaf Blight': 15, 'Peach Healthy': 16, 'Peach Bacterial Spot': 17, 'Potato Healthy': 18, 'Potato Early Blight': 19, 'Potato Late Blight': 20, 'Strawberry Healthy': 21, 'Strawberry Leaf Scorch': 22, 'Tomato Healthy': 23, 'Tomato Bacterial Spot': 24, 'Tomato Early Blight': 25, 'Tomato Late Blight': 26, 'Tomato Septoria Leaf Spot': 27, 'Tomato Yellow Leaf Curl Virus': 28 }

```
df = pd.read_csv('/content/drive/MyDrive/Research/TSAFOLDER/diseases.csv')
```

```
df = df[df['Path'].str.endswith((''.jpg', '.JPG'))]
```

```
train_df = df.sample(frac=0.8, random_state=42)
test_df = df.drop(train_df.index)
val_df = test_df.sample(frac=0.5, random_state=42)
test_df = test_df.drop(val_df.index)
```

```
train_df.head()
```



Path label

```
8364 /root/.cache/kagglehub/datasets/tushar5harma/p... 24
46929 /root/.cache/kagglehub/datasets/tushar5harma/p... 3
61351 /root/.cache/kagglehub/datasets/tushar5harma/p... 10
41550 /root/.cache/kagglehub/datasets/tushar5harma/p... 18
20692 /root/.cache/kagglehub/datasets/tushar5harma/p... 12
```

```
image = Image.open(train_df['Path'][1231])
image.show()
```

```
train_label = tf.keras.utils.to_categorical(train_df['label'], num_classes=29)
val_label = tf.keras.utils.to_categorical(val_df['label'], num_classes=29)
test_label = tf.keras.utils.to_categorical(test_df['label'], num_classes=29)
```

```
train_dataset = tf.data.Dataset.from_tensor_slices((train_df['Path'].values, train_label))
test_dataset = tf.data.Dataset.from_tensor_slices((test_df['Path'].values, test_label))
val_dataset = tf.data.Dataset.from_tensor_slices((val_df['Path'].values, val_label))
```

```
def load_image(image_path, label):
    image = tf.io.read_file(image_path)
    image = tf.image.decode_jpeg(image, channels=3) # Adjust for PNG if needed
    image = tf.image.resize(image, (224, 224))
    image = tf.cast(image, tf.float32) / 255.0
    return image, label
```

```
image,label = load_image(train_df['Path'][1231], train_label[1231])
```

label

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```
train_dataset = train_dataset.map(load_image).batch(32).shuffle(1000).prefetch(tf.data.AUTOTUNE)
test_dataset = test_dataset.map(load_image).batch(32).shuffle(1000).prefetch(tf.data.AUTOTUNE)
val_dataset = val_dataset.map(load_image).batch(32).shuffle(1000).prefetch(tf.data.AUTOTUNE)
```

```
base_model = tf.keras.applications.ResNet50(input_shape=(224,224,3), include_top=False, weights="imagenet")
```

```
base_model.trainable = True # Freeze layers
```

```
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(256, kernel_regularizer=regularizers.l2(0.01), activation="relu"),
    tf.keras.layers.Dense(29, kernel_regularizer=regularizers.l2(0.01), activation="softmax")
])
```

```
# Compile the model
```

```
model.compile(optimizer=optimizers.Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy', Precision(name='precision'), Recall(name='recall')])
```

```
model.summary()
```



Model: "sequential_4"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d_4 (GlobalAveragePooling2D)	(None, 2048)	0
dense_8 (Dense)	(None, 256)	524,544
dense_9 (Dense)	(None, 29)	7,453

Total params: 24,119,709 (92.01 MB)

Trainable params: 24,066,589 (91.81 MB)

```
checkpoint_filepath = '/content/drive/MyDrive/Research/TSAFOLDER/generalcrop.h5'
model_checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_best_only=True,
    monitor='val_accuracy', # Or another metric
    mode='max', # Or 'max' if monitoring accuracy, for example
    save_freq='epoch'
)
history = model.fit(
    train_dataset, # Replace with your training data
    validation_data=val_dataset, # Replace with your validation data
    epochs=5, # Adjust based on performance
    callbacks=[model_checkpoint_callback]
)
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_no_batch_normalization_tf_dim_ordering_tf_kernels_19578880/94765736 7s 0us/step

```
model.evaluate(test_dataset)
```

210/210 4s 15ms/step - accuracy: 0.9925 - loss: 0.0939 - precision: 0.9937 - recall: 0.9920
 [0.09685125946998596,
 0.9922515153884888,
 0.993728518486023,
 0.9916554689407349]

```
resnet = tf.keras.models.load_model("/content/drive/MyDrive/Research/TSAFOLDER/generalcrop.h5")
```

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you t

```
def load_image(image_path, label):
    image = tf.io.read_file(image_path)
    image = tf.image.decode_jpeg(image, channels=3) # Adjust for PNG if needed
    image = tf.image.resize(image, (224, 224))
    image = tf.cast(image, tf.float32) / 255.0
    return image, label
test_dataset = tf.data.Dataset.from_tensor_slices((test_df['Path'].values, test_label))
test_dataset = test_dataset.map(load_image).batch(1).prefetch(tf.data.AUTOTUNE)
```

```
resnet.evaluate(test_dataset)
```

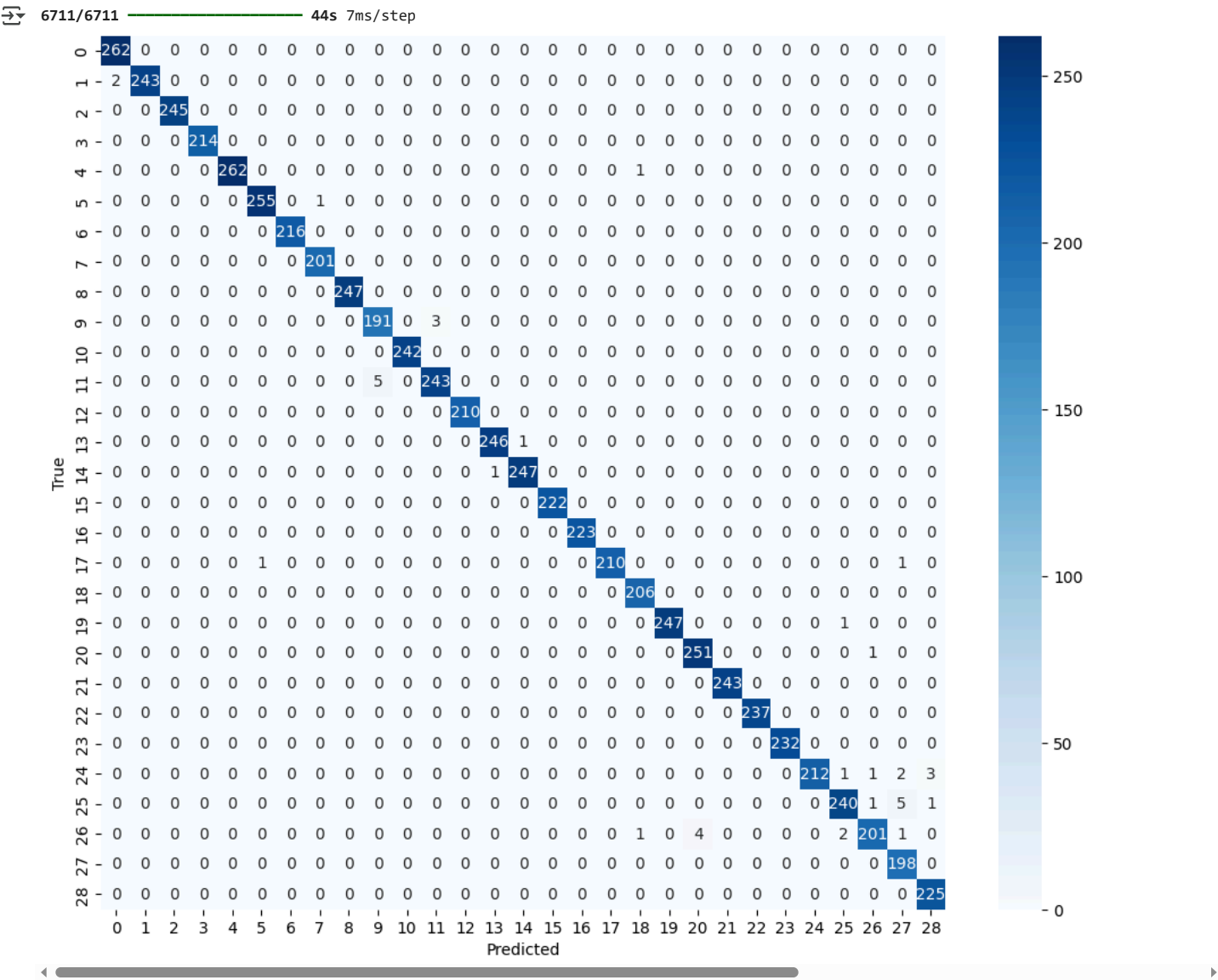
6711/6711 52s 7ms/step - accuracy: 0.9932 - loss: 0.1179 - precision: 0.9937 - recall: 0.9924
 [0.11727621406316757,
 0.9940396547317505,
 0.9944817423820496,
 0.9935926198959351]

```
from sklearn.metrics import confusion_matrix
```

```
y_true = np.argmax(test_label, axis=1) # Convert one-hot labels to class indices
y_pred = np.argmax(resnet.predict(test_dataset), axis=1)
```

```
conf_matrix = confusion_matrix(y_true, y_pred)
```

```
plt.figure(figsize=(12, 10))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```



Start coding or [generate](#) with AI.