```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("mexwell/crop-diseases-classification")

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

```
Path to dataset files: /kaggle/input/crop-diseases-classification
```

```
#/content/cassavabb.jpg


import numpy as np
import os
import PIL
import PIL.Image
import pandas as pd
import tensorflow as tf
import tensorflow_datasets as tfds
import PIL
from PIL import Image
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.metrics import Precision, Recall
import tensorflow.keras.regularizers as regularizers
from tensorflow.keras.utils import plot_model
import tensorflow as tf
from tensorflow.keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
import seaborn as sns


path = "/kaggle/input/crop-diseases-classification/versions/1"


df = pd.read_csv("/kaggle/input/crop-diseases-classification/Data/train.csv")


df.head()
```

	image_id	label
0	1000015157.jpg	0
1	1000201771.jpg	3
2	100042118.jpg	1
3	1000723321.jpg	1
4	1000812911.jpg	3

```
image_path = "/kaggle/input/crop-diseases-classification/Data/train_images"
image_list = os.listdir(image_path)



df = df[df["image_id"].isin(image_list)]
df.reset_index(drop=True, inplace=True)


path_name = image_path + "/"
df["image_path"] = df["image_id"].apply(lambda x: str(path_name+x))
df.head()
```

	image_id	label	image_path
0	157078263.jpg	3	/kaggle/input/crop-diseases-classification/Dat...
1	1574893536.jpg	3	/kaggle/input/crop-diseases-classification/Dat...
2	1575013487.jpg	3	/kaggle/input/crop-diseases-classification/Dat...
3	1576606254.jpg	0	/kaggle/input/crop-diseases-classification/Dat...
4	1579761476.jpg	1	/kaggle/input/crop-diseases-classification/Dat...

```
len(df)
```

```
17938
```

```
train_df = df[:13601]
val_df = df[13601:15301]
test_df = df[15301:17001]
```

```
train_label = tf.keras.utils.to_categorical(train_df['label'], num_classes=5)
val_label = tf.keras.utils.to_categorical(val_df['label'], num_classes=5)
test_label = tf.keras.utils.to_categorical(test_df['label'], num_classes=5)
```

```
test_df
```

	image_id	label	image_path
15301	505693686.jpg	3	/root/.cache/kagglehub/datasets/mexwell/crop-d...
15302	505820957.jpg	4	/root/.cache/kagglehub/datasets/mexwell/crop-d...
15303	50589657.jpg	1	/root/.cache/kagglehub/datasets/mexwell/crop-d...
15304	506080526.jpg	3	/root/.cache/kagglehub/datasets/mexwell/crop-d...
15305	506567755.jpg	3	/root/.cache/kagglehub/datasets/mexwell/crop-d...
...	...	...	...
16996	814122857.jpg	4	/root/.cache/kagglehub/datasets/mexwell/crop-d...
16997	814185128.jpg	3	/root/.cache/kagglehub/datasets/mexwell/crop-d...
16998	814288392.jpg	3	/root/.cache/kagglehub/datasets/mexwell/crop-d...
16999	814547184.jpg	1	/root/.cache/kagglehub/datasets/mexwell/crop-d...
17000	814722861.jpg	0	/root/.cache/kagglehub/datasets/mexwell/crop-d...

1700 rows × 3 columns

```
train_dataset = tf.data.Dataset.from_tensor_slices((train_df['image_path'].values, train_label))
test_dataset = tf.data.Dataset.from_tensor_slices((test_df['image_path'].values, test_label))
val_dataset = tf.data.Dataset.from_tensor_slices((val_df['image_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)
    return image, label
```

```
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)
```

```
len(train_dataset)
```

```
426
```

```
# Load base model
base_model = tf.keras.applications.ResNet50(input_shape=(600, 800, 3), include_top=False, weights="imagenet")

# Freeze initial layers
base_model.trainable = True

# Build the model
model = tf.keras.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.BatchNormalization(),
    layers.Dense(512, activation="relu"),
    layers.BatchNormalization(),
    layers.Dense(5, activation="softmax")  # No L2 regularization in the final layer
])

# Compile the model with AdamW optimizer
```

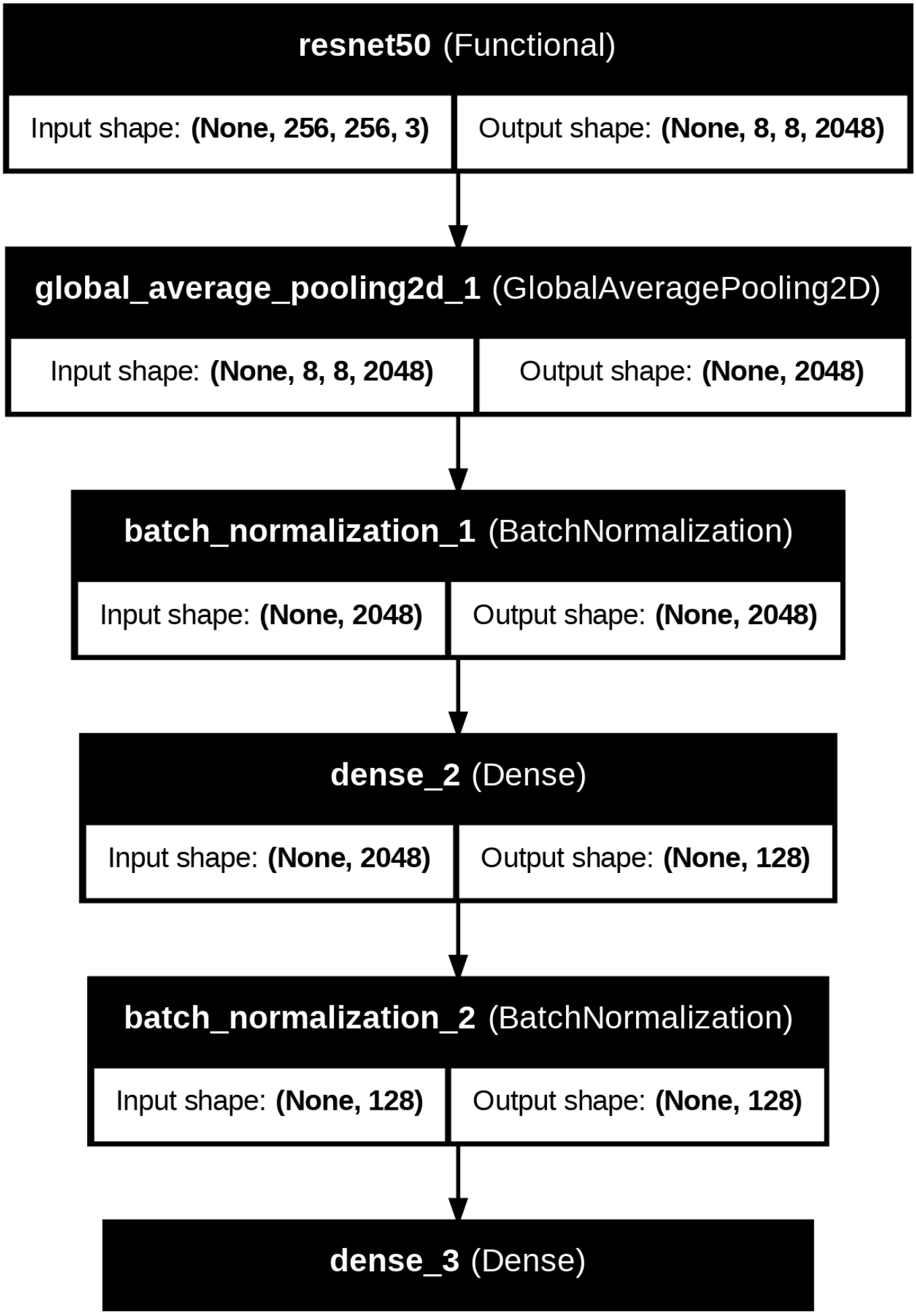
```
model.compile(optimizer=optimizers.AdamW(learning_rate=0.0001, weight_decay=1e-5),
              loss='categorical_crossentropy',
              metrics=['accuracy', Precision(name='precision'), Recall(name='recall')])

# Display model summary
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_nc
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
Model: "sequential"
```

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 19, 25, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
batch_normalization (BatchNormalization)	(None, 2048)	8,192
dense (Dense)	(None, 512)	1,049,088
batch_normalization_1 (BatchNormalization)	(None, 512)	2,048
dense_1 (Dense)	(None, 5)	2,565

```
Total params: 24,649,605 (94.03 MB)
```

```
plot_model(model, to_file='model_diagram.png', show_shapes=True, show_layer_names=True)
```


resnet50 (Functional)
Input shape: (None, 256, 256, 3)
Output shape: (None, 8, 8, 2048)
global_average_pooling2d_1 (GlobalAveragePooling2D)
Input shape: (None, 8, 8, 2048)
Output shape: (None, 2048)
batch_normalization_1 (BatchNormalization)
Input shape: (None, 2048)
Output shape: (None, 2048)
dense_2 (Dense)
Input shape: (None, 2048)
Output shape: (None, 128)
batch_normalization_2 (BatchNormalization)
Input shape: (None, 128)
Output shape: (None, 128)
dense_3 (Dense)

Input shape: **(None, 128)**	Output shape: **(None, 5)**

```
checkpoint_filepath = '/content/drive/MyDrive/Research/TSAFOLDER/crop.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]
)
```

```
Epoch 1/5
426/426 ━━━━━━━━━━━━━━━━━━━━ 0s 323ms/step - accuracy: 0.7048 - loss: 0.9617 - precision: 0.7584 - recall: 0.6560WARNING:absl:You are sa
426/426 ━━━━━━━━━━━━━━━━━━━━ 263s 371ms/step - accuracy: 0.7050 - loss: 0.9612 - precision: 0.7585 - recall: 0.6562 - val_accuracy: 0.83
Epoch 2/5
426/426 ━━━━━━━━━━━━━━━━━━━━ 127s 284ms/step - accuracy: 0.8874 - loss: 0.3399 - precision: 0.9086 - recall: 0.8687 - val_accuracy: 0.82
Epoch 3/5
426/426 ━━━━━━━━━━━━━━━━━━━━ 0s 272ms/step - accuracy: 0.9514 - loss: 0.1528 - precision: 0.9574 - recall: 0.9440WARNING:absl:You are sa
426/426 ━━━━━━━━━━━━━━━━━━━━ 130s 293ms/step - accuracy: 0.9514 - loss: 0.1528 - precision: 0.9574 - recall: 0.9440 - val_accuracy: 0.84
Epoch 4/5
426/426 ━━━━━━━━━━━━━━━━━━━━ 127s 284ms/step - accuracy: 0.9790 - loss: 0.0727 - precision: 0.9814 - recall: 0.9760 - val_accuracy: 0.83
Epoch 5/5
426/426 ━━━━━━━━━━━━━━━━━━━━ 126s 284ms/step - accuracy: 0.9735 - loss: 0.0784 - precision: 0.9773 - recall: 0.9696 - val_accuracy: 0.82
```

Start coding or generate with AI.

Double-click (or enter) to edit

```
model.save("/content/drive/MyDrive/Research/TSAFOLDER/disease_model2.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 consi
```

```
model = tf.keras.models.load_model("/content/drive/MyDrive/Research/TSAFOLDER/crop.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
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 19, 25, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
batch_normalization (BatchNormalization)	(None, 2048)	8,192
dense (Dense)	(None, 512)	1,049,088
batch_normalization_1 (BatchNormalization)	(None, 512)	2,048
dense_1 (Dense)	(None, 5)	2,565

Trainable params: 24,591,365 (93.81 MB)

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

```
54/54 ━━━━━━━━━━━━━━━━━━━━ 12s 108ms/step - accuracy: 0.8527 - loss: 0.5308 - precision: 0.8659 - recall: 0.8392
[0.5145025849342346,
 0.8617647290229797,
 0.8759124279022217,
 0.8470588326454163]
```

```
preds = model.predict(test_dataset)
preds = np.argmax(preds, axis=1)
```

```
1700/1700 ━━━━━━━━━━━━━━━━━━━━ 17s 8ms/step
```

```
preds
```

```
array([3, 2, 1, ..., 3, 1, 0])
```

```
true_values = np.argmax(test_label, axis=1)
true_values
```

```
array([3, 4, 1, ..., 3, 1, 0])
```

```
confusion_matrix = tf.math.confusion_matrix(true_values, preds)
```

```
# Assuming true_values and preds are your true labels and predicted labels
confusion_matrix = tf.math.confusion_matrix(true_values, preds)

# Convert confusion matrix to numpy array for easier handling
confusion_matrix = confusion_matrix.numpy()

# Create a heatmap for visualization
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```