```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Create a list of the directories containing your training, validation, and testing datasets:
training_dir = '/content/drive/MyDrive/dataset/train'
validation_dir = '/content/drive/MyDrive/dataset/validation'
testing_dir = '/content/drive/MyDrive/dataset/test'

# Create a list of the labels for each class in your dataset:
classes = ['apple','banana','beetroot','bell pepper','cabbage','capsicum','carrot','cauliflower','chilli pepper','corn','cucumber','eggplant'
```

```
# Create a generator for the training dataset:
train_generator = keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    fill_mode='nearest'
)

train_data = train_generator.flow_from_directory(
    training_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=True
)
```

```
Found 3055 images belonging to 36 classes.
```

```
# Create a generator for the validation dataset:
validation_generator = keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255
)

validation_data = validation_generator.flow_from_directory(
    validation_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 351 images belonging to 36 classes.
```

```
# Create a generator for the testing dataset:
testing_generator = keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255
)

testing_data = testing_generator.flow_from_directory(
    testing_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 359 images belonging to 36 classes.
```

```
# Define the model architecture:
model = Sequential()

model.add(Conv2D(32, (3, 3), padding='same',
          activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2), strides=(2, 2)))

model.add(Conv2D(54, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D((2, 2), strides=(2, 2)))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(MaxPooling2D((2, 2), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(len(classes), activation='softmax'))


# Compile the model:
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])


# # Train the model:
model.fit(
    train_data,
    steps_per_epoch=len(train_data),
    epochs=15,
    validation_data=validation_data,
    validation_steps=len(validation_data)
)
```

```
/usr/local/lib/python3.10/dist-packages/PIL/Image.py:975: UserWarning: Palette images with Transparency expressed in bytes should be cor
  warnings.warn(
Epoch 1/15
96/96 [==============================] - 933s 10s/step - loss: 3.6510 - accuracy: 0.0370 - val_loss: 3.4244 - val_accuracy: 0.0883
Epoch 2/15
96/96 [==============================] - 172s 2s/step - loss: 3.1671 - accuracy: 0.1267 - val_loss: 2.4430 - val_accuracy: 0.3533
Epoch 3/15
96/96 [==============================] - 157s 2s/step - loss: 2.3558 - accuracy: 0.3247 - val_loss: 1.3924 - val_accuracy: 0.6325
Epoch 4/15
96/96 [==============================] - 153s 2s/step - loss: 1.5113 - accuracy: 0.5591 - val_loss: 0.7026 - val_accuracy: 0.8547
Epoch 5/15
96/96 [==============================] - 152s 2s/step - loss: 0.6712 - accuracy: 0.8137 - val_loss: 0.4910 - val_accuracy: 0.9259
Epoch 6/15
96/96 [==============================] - 145s 2s/step - loss: 0.3425 - accuracy: 0.9201 - val_loss: 0.3642 - val_accuracy: 0.9430
Epoch 7/15
96/96 [==============================] - 150s 2s/step - loss: 0.2233 - accuracy: 0.9538 - val_loss: 0.3242 - val_accuracy: 0.9630
Epoch 8/15
96/96 [==============================] - 144s 2s/step - loss: 0.1362 - accuracy: 0.9781 - val_loss: 0.4697 - val_accuracy: 0.9516
Epoch 9/15
96/96 [==============================] - 143s 1s/step - loss: 0.0865 - accuracy: 0.9872 - val_loss: 0.3668 - val_accuracy: 0.9601
Epoch 10/15
96/96 [==============================] - 146s 2s/step - loss: 0.0822 - accuracy: 0.9866 - val_loss: 0.2985 - val_accuracy: 0.9630
Epoch 11/15
96/96 [==============================] - 144s 2s/step - loss: 0.0728 - accuracy: 0.9892 - val_loss: 0.3097 - val_accuracy: 0.9601
Epoch 12/15
96/96 [==============================] - 144s 1s/step - loss: 0.0575 - accuracy: 0.9905 - val_loss: 0.3548 - val_accuracy: 0.9601
Epoch 13/15
96/96 [==============================] - 145s 2s/step - loss: 0.0473 - accuracy: 0.9908 - val_loss: 0.3341 - val_accuracy: 0.9601
Epoch 14/15
96/96 [==============================] - 143s 2s/step - loss: 0.0421 - accuracy: 0.9895 - val_loss: 0.3728 - val_accuracy: 0.9544
Epoch 15/15
96/96 [==============================] - 147s 2s/step - loss: 0.0501 - accuracy: 0.9892 - val_loss: 0.3283 - val_accuracy: 0.9573
<keras.callbacks.History at 0x7fc83cb84820>
```

```
# # Evaluate the model:
score = model.evaluate(
    testing_data,
    steps=len(testing_data)
)

print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
12/12 [==============================] - 96s 9s/step - loss: 0.3211 - accuracy: 0.9582
Test loss: 0.32107415795326233
Test accuracy: 0.9582172632217407
```

```
# # Save the model:
model.save('model3.h5')


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896
 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 54)      15606

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 54)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 64)        31168

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 flatten (Flatten)           (None, 50176)             0

 dense (Dense)               (None, 128)               6422656

 dense_1 (Dense)             (None, 36)                4644

=================================================================
Total params: 6,474,970
Trainable params: 6,474,970
Non-trainable params: 0
_________________________________________________________________
```