Vegetable image classifier using MobileNetV3

In [1]:
```
#import data by kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

In [2]:
```
!kaggle datasets download -d misrakahmed/vegetable-image-dataset
```

```
Warning: Your Kaggle API key is readable by other users on this sy
stem! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.jso
n'
Downloading vegetable-image-dataset.zip to /content
 97% 519M/534M [00:05<00:00, 69.4MB/s]
100% 534M/534M [00:05<00:00, 101MB/s]
```

In [3]:
```
#file unzip
import zipfile
zip_ref = zipfile.ZipFile('/content/vegetable-image-dataset.zip')
zip_ref.extractall('/content')
zip_ref.close()
```

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import os
import PIL
import pathlib
import tensorflow as tf
from tensorflow import keras
```

In [8]:
```
train_dir = pathlib.Path('/content/Vegetable Images/train')
val_dir = pathlib.Path('/content/Vegetable Images/validation')
test_dir = pathlib.Path('/content/Vegetable Images/test')
train_image_count = len(list(train_dir.glob('*/*.jpg')))
train_image_count
```

Out[8]:
```
15000
```

In [9]:
```
batch_size = 32
image_width = 224
image_height = 224

train_ds = keras.preprocessing.image_dataset_from_directory( train_
  seed=123,
  #validation_split=0.5, #subset='training',
  image_size=(image_height, image_width),
  batch_size=batch_size
)
val_ds = keras.preprocessing.image_dataset_from_directory( val_dir,
  seed=123,
  image_size=(image_height, image_width),
  batch_size=batch_size
)

test_ds = keras.preprocessing.image_dataset_from_directory( test_di
  seed=123,
  image_size=(image_height, image_width),
  batch_size=batch_size
)
```

```
Found 15000 files belonging to 15 classes.
Found 3000 files belonging to 15 classes.
Found 3000 files belonging to 15 classes.
```

In [10]:
```
#check data class
class_names = train_ds.class_names
class_names
```

Out[10]:
```
['Bean',
 'Bitter_Gourd',
 'Bottle_Gourd',
 'Brinjal',
 'Broccoli',
 'Cabbage',
 'Capsicum',
 'Carrot',
 'Cauliflower',
 'Cucumber',
 'Papaya',
 'Potato',
 'Pumpkin',
 'Radish',
 'Tomato']
```

In [11]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i+1)
    plt.imshow(images[i].numpy().astype('uint8'))
    plt.title(class_names[labels[i]])
    plt.axis('off')
```

In [12]:
```
# Criando o modelo base em cima do modelo MobileNetV3
base_model = keras.applications.MobileNetV3Small(input_shape=(image
                                                  classes=400,
                                                  include_top=False,
                                                  weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/ke
ras-applications/mobilenet_v3/weights_mobilenet_v3_small_224_1.0_f
loat_no_top_v2.h5
(https://storage.googleapis.com/tensorflow/keras-applications/mobi
lenet_v3/weights_mobilenet_v3_small_224_1.0_float_no_top_v2.h5)
4334752/4334752 [==============================] - 0s 0us/step
```

In [13]:
```
# Freeze convolutional base
base_model.trainable = False
base_model.summary()
```

```
Model: "MobilenetV3small"
_________________________________________________________________
_________________________________
 Layer (type)                   Output Shape         Param #     C
onnected to
==================================================================
================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           [
]
                                )]

 rescaling (Rescaling)          (None, 224, 224, 3)  0           [
'input_1[0][0]']

 Conv (Conv2D)                  (None, 112, 112, 16  432         [
'rescaling[0][0]']
                                )

 Conv/BatchNorm (BatchNormaliza  (None, 112, 112, 16  64         [
'Conv[0][0]']
```

In [14]:
```
data_augmentation = keras.models.Sequential([
  keras.layers.RandomFlip('horizontal'),
  keras.layers.RandomRotation(0.2)
])
```

In [15]:
```
num_classes = len(class_names)# 7

inputs = keras.Input(shape=(image_width, image_height, 3))
#x = data_augmentation(inputs)
x = keras.applications.mobilenet_v3.preprocess_input(inputs)
x = base_model(x, training=False)
x = keras.layers.GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)


outputs = keras.layers.Dense(num_classes, activation='softmax')(x)
model = keras.Model(inputs, outputs)
```

In [16]:
```
#compile model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 MobilenetV3small (Functiona  (None, 7, 7, 576)        939120
 l)

 global_average_pooling2d (G  (None, 576)              0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 576)               0

 dense (Dense)               (None, 15)                8655

=================================================================
Total params: 947,775
Trainable params: 8,655
Non-trainable params: 939,120
_________________________________________________________________
```

In [17]:
```
#fitting model
initial_epochs = 10

early_stop = keras.callbacks.EarlyStopping(patience=1, restore_best

history = model.fit(train_ds,
          validation_data=val_ds,
          epochs=initial_epochs,
          callbacks=[early_stop])
```

```
Epoch 1/10
469/469 [==============================] - 32s 46ms/step - loss: 0
.4650 - accuracy: 0.8921 - val_loss: 0.0958 - val_accuracy: 0.9850
Epoch 2/10
469/469 [==============================] - 21s 44ms/step - loss: 0
.0846 - accuracy: 0.9876 - val_loss: 0.0458 - val_accuracy: 0.9930
Epoch 3/10
469/469 [==============================] - 21s 44ms/step - loss: 0
.0500 - accuracy: 0.9924 - val_loss: 0.0305 - val_accuracy: 0.9947
Epoch 4/10
469/469 [==============================] - 21s 44ms/step - loss: 0
.0345 - accuracy: 0.9945 - val_loss: 0.0225 - val_accuracy: 0.9960
Epoch 5/10
469/469 [==============================] - 21s 44ms/step - loss: 0
.0275 - accuracy: 0.9951 - val_loss: 0.0181 - val_accuracy: 0.9963
Epoch 6/10
469/469 [==============================] - 20s 42ms/step - loss: 0
.0211 - accuracy: 0.9970 - val_loss: 0.0156 - val_accuracy: 0.9963
Epoch 7/10
469/469 [==============================] - 21s 44ms/step - loss: 0
.0167 - accuracy: 0.9979 - val_loss: 0.0131 - val_accuracy: 0.9973
Epoch 8/10
469/469 [==============================] - 20s 42ms/step - loss: 0
.0139 - accuracy: 0.9979 - val_loss: 0.0106 - val_accuracy: 0.9977
Epoch 9/10
469/469 [==============================] - 21s 44ms/step - loss: 0
.0123 - accuracy: 0.9978 - val_loss: 0.0106 - val_accuracy: 0.9983
Epoch 10/10
469/469 [==============================] - 20s 42ms/step - loss: 0
.0107 - accuracy: 0.9984 - val_loss: 0.0096 - val_accuracy: 0.9977
```

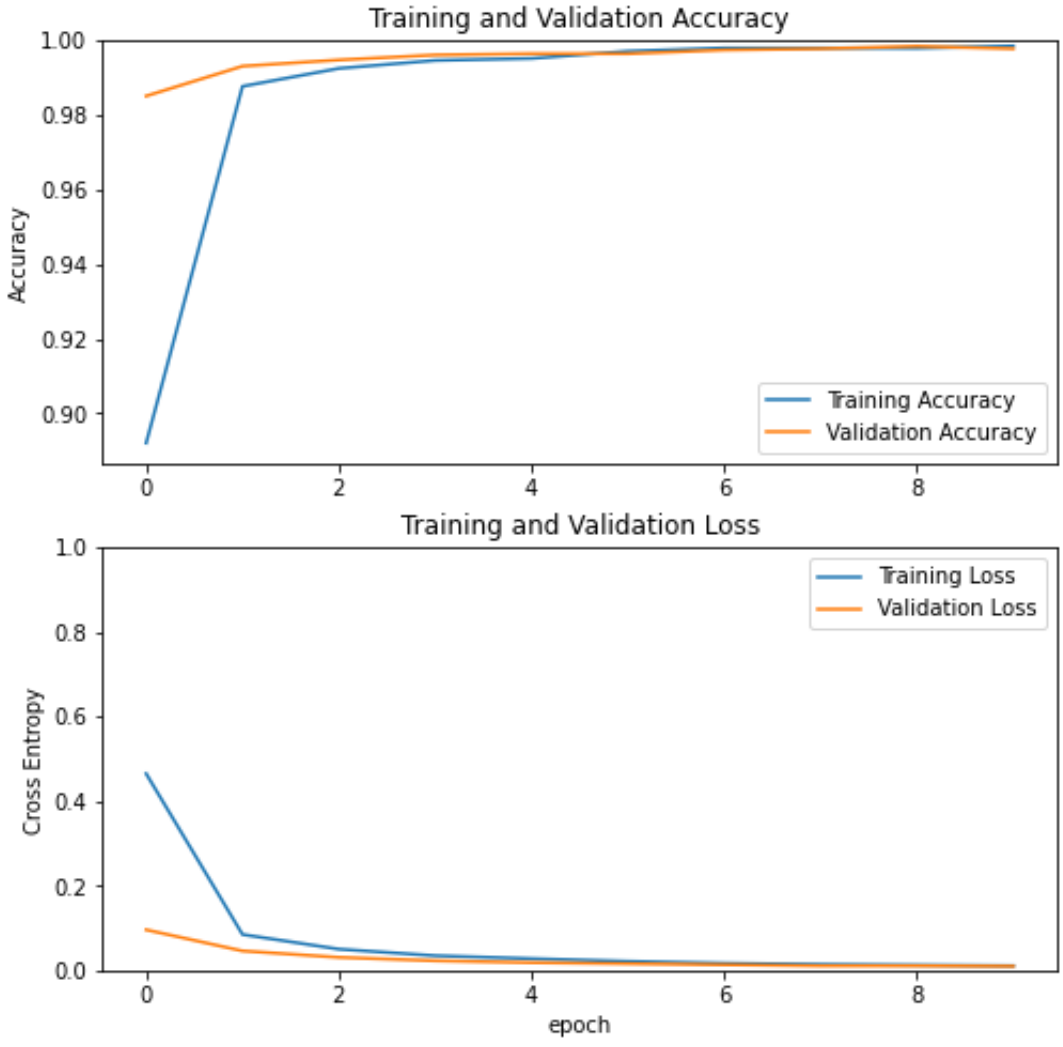
In [18]:
```
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```



In [19]:
```
plt.figure(figsize=(10, 100))
for images, labels in test_ds.take(1):
    prediction = model.predict(images,batch_size=32)
    for i in range(9):
        ax = plt.subplot(9, 1, i+1)
        pred = np.argmax(prediction[i])
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(f'Predito: {class_names[pred]} - Real: {class_nam
        plt.axis('off')
```

```
1/1 [==============================] - 1s 1s/step
```

In [20]:
```
#Visualize the result
results = model.evaluate(test_ds, verbose=0)
```

In [23]:
```
print(" Test Loss: {:.5f}".format(results[0]))
print(" Accuracy on the test set: {:.2f}%".format(results[1] * 100)
```

```
 Test Loss: 0.00997
 Accuracy on the test set: 99.80%
```

In [24]:
```
test_ds_unbatch = tf.keras.utils.image_dataset_from_directory(
    test_dir,
    seed=None,
    shuffle=False,
    image_size=(image_height, image_width),
    batch_size=image_height,
)
images = list(test_ds_unbatch.map(lambda x, y: x))
labels = list(test_ds_unbatch.map(lambda x, y: y))
```

```
Found 3000 files belonging to 15 classes.
```

In [25]:
```
y_test = np.concatenate([y for x, y in test_ds_unbatch], axis=0)
prediction = model.predict(test_ds_unbatch)
pred = np.argmax(prediction,axis=1)
print(pred)
from sklearn.metrics import accuracy_score

accuracy_score(pred,y_test)
```

```
14/14 [==============================] - 5s 257ms/step
[ 0  0  0 ... 14 14 14]
```

Out[25]:
```
0.998
```

In [27]:
```
#save model
model.save('/content/drive/MyDrive/Colab Notebooks/Vagitable.h5')
```

In [ ]:
```

```