```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

```
train_dir = "/content/drive/MyDrive/Potato/Train"
test_dir = "/content/drive/MyDrive/Potato/Test"
valid_dir = "/content/drive/MyDrive/Potato/Valid"
```

```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.experimental.preprocessing.RandomFlip("horizontal",input_shape=(224, 224, 3)),
  tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),
  tf.keras.layers.experimental.preprocessing.RandomZoom(0.2),
  tf.keras.layers.experimental.preprocessing.RandomHeight(0.2),
  tf.keras.layers.experimental.preprocessing.RandomWidth(0.2),
  tf.keras.layers.experimental.preprocessing.Rescaling(1./255)
], name ="data_augmentation")
```

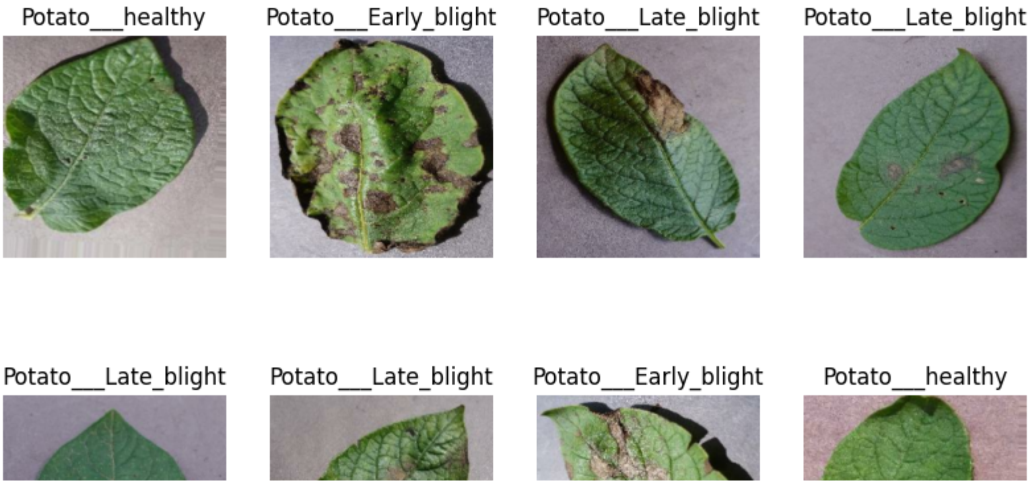
```
import tensorflow as tf

IMG_SIZE = (224, 224)
BATCH_SIZE = 32
train_data = tf.keras.preprocessing.image_dataset_from_directory(
    directory = train_dir,
    image_size = IMG_SIZE,
    label_mode = 'categorical',
    batch_size = BATCH_SIZE,
    shuffle = True
).cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

test_data = tf.keras.preprocessing.image_dataset_from_directory(
    directory = test_dir,
    image_size = IMG_SIZE,
    label_mode = 'categorical',
    batch_size = BATCH_SIZE
).cache().prefetch(buffer_size=tf.data.AUTOTUNE)

valid_datasets = tf.keras.preprocessing.image_dataset_from_directory(
    directory = valid_dir,
    image_size = IMG_SIZE,
    label_mode = 'categorical',
    batch_size = BATCH_SIZE
)

class_names = valid_datasets.class_names
valid_data = valid_datasets.cache().prefetch(buffer_size=tf.data.AUTOTUNE)
```

```
     Found 900 files belonging to 3 classes.
     Found 300 files belonging to 3 classes.
     Found 300 files belonging to 3 classes.
```

```
[(images, label_batch)] = train_data.take(1)
total_classes = label_batch.shape[-1]
total_classes
```

```
     3
```

```
plt.figure(figsize=(10,10))
for image_batch,label_batch in train_data.take(1):
    for i in range(0,8):
        plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint32"))
        plt.title(class_names[np.argmax(label_batch[i])])
        plt.axis("off")
```



```
model = tf.keras.Sequential([
    data_augmentation,
    tf.keras.layers.Conv2D(60, 3, activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2),
    tf.keras.layers.Conv2D(60, 3, activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2),
    tf.keras.layers.Conv2D(60, 3, activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2),
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(total_classes, activation='softmax', name='output_layers')

])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 data_augmentation (Sequenti  (None, None, None, 3)    0         
 al)                                                             
                                                                 
 conv2d (Conv2D)             (None, None, None, 60)    1680      
                                                                 
 max_pooling2d (MaxPooling2D  (None, None, None, 60)   0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, None, None, 60)    32460     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, None, None, 60)   0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, None, None, 60)    32460     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, None, None, 60)   0         
 2D)                                                             
                                                                 
 global_average_pooling2d (G  (None, 60)               0         
 lobalAveragePooling2D)                                          
                                                                 
 output_layers (Dense)       (None, 3)                 183       
                                                                 
=================================================================
Total params: 66,783
Trainable params: 66,783
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(
    loss = 'categorical_crossentropy',
    optimizer = tf.keras.optimizers.Adam(),
    metrics = ['accuracy']
)
```

```
checkpoint_path = "/CheckPoint/cp.ckpt"
checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    checkpoint_path,
    save_weights_only=True,
    monitor='val_accuracy',
    save_best_only=True
)
```

```
history = model.fit(
history = model.fit(
    train_data,
    epochs=5,
    validation_data = test_data,
    validation_steps = len(test_data),
    callbacks = [
        checkpoint_callback,
    ]
)
```

```
Epoch 1/5
29/29 [==============================] - 301s 10s/step - loss: 1.0703 - accuracy: 0.4111 - val_loss: 0.9922 - val_accuracy: 0.3700
Epoch 2/5
29/29 [==============================] - 192s 7s/step - loss: 0.9226 - accuracy: 0.5444 - val_loss: 0.7869 - val_accuracy: 0.6400
Epoch 3/5
29/29 [==============================] - 189s 7s/step - loss: 0.8201 - accuracy: 0.6122 - val_loss: 0.7526 - val_accuracy: 0.6133
Epoch 4/5
29/29 [==============================] - 191s 7s/step - loss: 0.7775 - accuracy: 0.6456 - val_loss: 0.6472 - val_accuracy: 0.7500
Epoch 5/5
29/29 [==============================] - 194s 7s/step - loss: 0.6173 - accuracy: 0.7322 - val_loss: 0.5162 - val_accuracy: 0.8400
```

```
def bestWeightModelEvaluate(model, weight_path, data):
  cp_model = tf.keras.models.clone_model(model)
  cp_model.compile(
      loss = tf.keras.losses.CategoricalCrossentropy(),
      optimizer = tf.keras.optimizers.Adam(),
      metrics = ['accuracy']
  )
  cp_model.load_weights(weight_path)
  cp_model.evaluate(data)
```

```
bestWeightModelEvaluate(
    model = model,
    weight_path= checkpoint_path,
    data = valid_data
)
```

```
10/10 [==============================] - 44s 1s/step - loss: 0.5612 - accuracy: 0.7900
```

```
bestWeightModelEvaluate(
    model = model,
    weight_path= checkpoint_path,
    data = test_data
)
```

```
10/10 [==============================] - 16s 2s/step - loss: 0.5162 - accuracy: 0.8400
```

```
! wget "https://github.com/harshini22-hue/Potato-plant-disease-detection/blob/main/pretrain_model.h5"
```

```
--2023-05-11 18:22:06--  https://github.com/harshini22-hue/Potato-plant-disease-detection/blob/main/pretrain_model.h5
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘pretrain_model.h5’

pretrain_model.h5       [ <=>                ] 145.49K  --.-KB/s    in 0.04s

2023-05-11 18:22:06 (3.90 MB/s) - ‘pretrain_model.h5’ saved [148978]
```

```
!pip install h5py
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (3.8.0)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.10/dist-packages (from h5py) (1.22.4)
```

```
import h5py as h5
```

```
model.save('/content/pretrain_model.h5')
```

```
Load_model = tf.keras.models.load_model('pretrain_model.h5')
```

```
Load_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 data_augmentation (Sequenti  (None, None, None, 3)    0
 al)

 conv2d (Conv2D)             (None, None, None, 60)    1680

 max_pooling2d (MaxPooling2D  (None, None, None, 60)   0
 )

 conv2d_1 (Conv2D)           (None, None, None, 60)    32460

 max_pooling2d_1 (MaxPooling  (None, None, None, 60)   0
 2D)

 conv2d_2 (Conv2D)           (None, None, None, 60)    32460

 max_pooling2d_2 (MaxPooling  (None, None, None, 60)   0
 2D)

 global_average_pooling2d (G  (None, 60)               0
 lobalAveragePooling2D)

 output_layers (Dense)       (None, 3)                 183

=================================================================
Total params: 66,783
Trainable params: 66,783
Non-trainable params: 0
_________________________________________________________________
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 data_augmentation (Sequenti  (None, None, None, 3)    0
 al)

 conv2d (Conv2D)             (None, None, None, 60)    1680

 max_pooling2d (MaxPooling2D  (None, None, None, 60)   0
 )

 conv2d_1 (Conv2D)           (None, None, None, 60)    32460

 max_pooling2d_1 (MaxPooling  (None, None, None, 60)   0
 2D)

 conv2d_2 (Conv2D)           (None, None, None, 60)    32460

 max_pooling2d_2 (MaxPooling  (None, None, None, 60)   0
 2D)

 global_average_pooling2d (G  (None, 60)               0
 lobalAveragePooling2D)

 output_layers (Dense)       (None, 3)                 183

=================================================================
Total params: 66,783
Trainable params: 66,783
Non-trainable params: 0
_________________________________________________________________
```

Evaluate load model with valid data and test_data

```
Load_model.evaluate(test_data)
```

```
10/10 [==============================] - 13s 1s/step - loss: 0.5162 - accuracy: 0.8400
[0.5161631107330322, 0.8399999737739563]
```

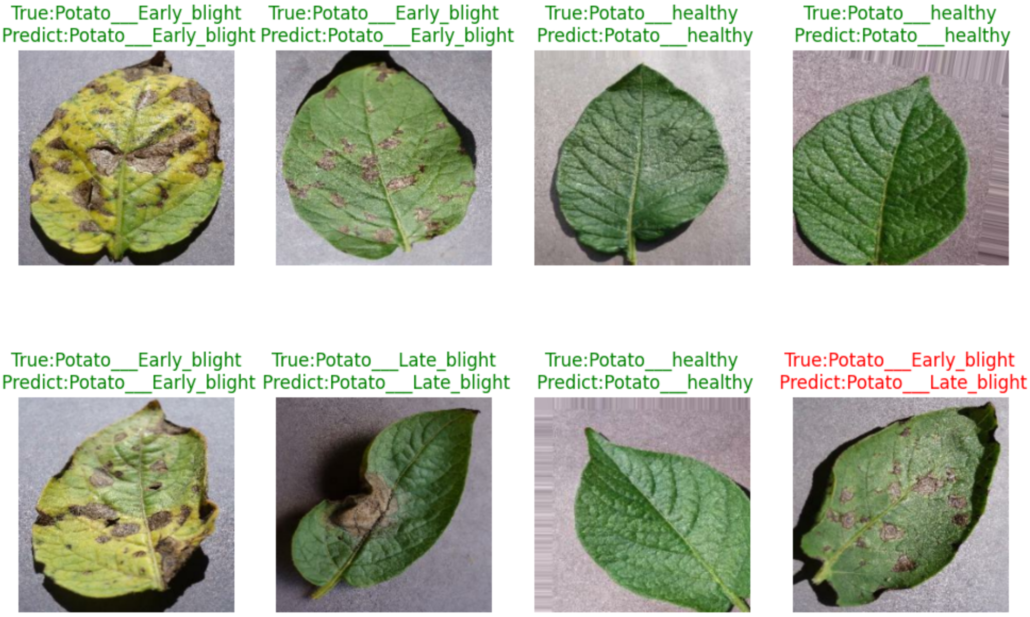
```
Load_model.evaluate(valid_data)
```

```
10/10 [==============================] - 14s 1s/step - loss: 0.5612 - accuracy: 0.7900
[0.5612201690673828, 0.7900000214576721]
```

```
import numpy as np
predictions = Load_model.predict(test_data)
predictions = np.argmax(predictions, axis=-1)
predictions.shape
```

```
10/10 [==============================] - 14s 1s/step
(300,)
```

```
plt.figure(figsize=(12,12))
for image_batch,label_batch in test_data.take(1):
    for i in range(0,8):
        plt.subplot(3,4,i+1)
        plt.imshow(image_batch[i].numpy().astype("uint32"))
        true_class = class_names[np.argmax(label_batch[i])]
        predict_class = class_names[predictions[i]]
        title = f"""True:{true_class}\n Predict:{predict_class}"""
        plt.title(title, color='g' if true_class==predict_class else 'r')
        plt.axis("off")
```



True:Potato___Early_blight
Predict:Potato___Early_blight

True:Potato___Early_blight
Predict:Potato___Early_blight

True:Potato___healthy
Predict:Potato___healthy

True:Potato___healthy
Predict:Potato___healthy

True:Potato___Early_blight
Predict:Potato___Early_blight

True:Potato___Late_blight
Predict:Potato___Late_blight

True:Potato___healthy
Predict:Potato___healthy

True:Potato___Early_blight
Predict:Potato___Late_blight

RESNET

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import accuracy_score
```

```
train_datagen = ImageDataGenerator(rescale = 1./255., rotation_range = 40, width_shift_range = 0.2, height_shift_range = 0.2, shear_rang

test_datagen = ImageDataGenerator(rescale = 1.0/255.)

train_generator = train_datagen.flow_from_directory(train_dir, batch_size = 20, class_mode = 'binary', target_size = (224, 224))

validation_generator = test_datagen.flow_from_directory( valid_dir, batch_size = 20, class_mode = 'binary', target_size = (224, 224))
```

```
Found 900 images belonging to 3 classes.
Found 300 images belonging to 3 classes.
```

```
from tensorflow.keras.applications import ResNet50

base_model = ResNet50(input_shape=(224, 224,3), include_top=False, weights="imagenet")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kerne
94765736/94765736 [==============================] - 1s 0us/step
```

```
for layer in base_model.layers:
    layer.trainable = False
```

```
from tensorflow.keras.applications import ResNet50
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Flatten, GlobalAveragePooling2D
```

```
base_model = Sequential()
base_model.add(ResNet50(include_top=False, weights='imagenet', pooling='max'))
base_model.add(Dense(1, activation='sigmoid'))
```

```
base_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
resnet_history = base_model.fit(train_generator, validation_data = validation_generator, steps_per_epoch = 10, epochs = 10)
```

```
Epoch 1/10
10/10 [==============================] - ETA: 0s - loss: 0.0000e+00 - accuracy: 0.5900
---------------------------------------------------------------------------
KeyboardInterrupt                          Traceback (most recent call last)
<ipython-input-37-38c27c1074d6> in <cell line: 1>()
----> 1 resnet_history = base_model.fit(train_generator, validation_data = validation_generator,
steps_per_epoch = 10, epochs = 10)

                                    8 frames

/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name,
num_outputs, inputs, attrs, ctx, name)
     50   try:
     51     ctx.ensure_initialized()
---> 52     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     53                                         inputs, attrs, num_outputs)
     54   except core._NotOkStatusException as e:

KeyboardInterrupt:
```

VGG-16

```
train_datagen = ImageDataGenerator(rescale = 1./255.,rotation_range = 40, width_shift_range = 0.2, height_shift_range = 0.2, shear_range
test_datagen = ImageDataGenerator( rescale = 1.0/255. )
```

```
train_generator = train_datagen.flow_from_directory(train_dir, batch_size = 20, class_mode = 'binary', target_size = (224, 224))

# Flow validation images in batches of 20 using test_datagen generator
validation_generator = test_datagen.flow_from_directory( valid_dir,  batch_size = 20, class_mode = 'binary', target_size = (224, 224))
```

```
Found 900 images belonging to 3 classes.
Found 300 images belonging to 3 classes.
```

Creating Base Model

```
from tensorflow.keras.applications.vgg16 import VGG16

base_model = VGG16(input_shape = (224, 224, 3), # Shape of our images
include_top = False, # Leave out the last fully connected layer
weights = 'imagenet')
for layer in base_model.layers:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_r
58889256/58889256 [==============================] - 0s 0us/step
```

```
from tensorflow.keras import layers
```

```
# Flatten the output layer to 1 dimension
x = layers.Flatten()(base_model.output)

# Add a fully connected layer with 512 hidden units and ReLU activation
x = layers.Dense(512, activation='relu')(x)

# Add a dropout rate of 0.5
x = layers.Dropout(0.5)(x)

# Add a final sigmoid layer with 1 node for classification output
x = layers.Dense(1, activation='sigmoid')(x)

model = tf.keras.models.Model(base_model.input, x)
```

```
base_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
vgg = model.fit(train_generator, validation_data = validation_generator, steps_per_epoch = 10, epochs = 10)
```

```
Epoch 1/10
10/10 [==============================] - 323s 34s/step - loss: -6.3329 - acc: 0.3850 - val_loss: -17.0681 - val_acc: 0.3333
Epoch 2/10
10/10 [==============================] - 333s 35s/step - loss: -18.6839 - acc: 0.3500 - val_loss: -35.6076 - val_acc: 0.3733
Epoch 3/10
10/10 [==============================] - 316s 34s/step - loss: -30.4446 - acc: 0.3500 - val_loss: -52.7398 - val_acc: 0.3733
Epoch 4/10
10/10 [==============================] - 331s 35s/step - loss: -47.6787 - acc: 0.3750 - val_loss: -73.4731 - val_acc: 0.3667
Epoch 5/10
10/10 [==============================] - 330s 35s/step - loss: -54.0805 - acc: 0.3900 - val_loss: -93.3878 - val_acc: 0.4300
Epoch 6/10
 7/10 [====================>.........] - ETA: 38s - loss: -113.6100 - acc: 0.3429
```

▸ Executing (30m 40s) <cell line: 1> › error_handler() › fit() › error_handler() › __call__() › _call() › __call__() › _call_flat() › call() › quick_execute() ··· ✕