Loading the Packages

In [13]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import sys
import os
from keras.applications.vgg16 import VGG16
import keras
from numpy import load
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from keras import backend
from keras.layers import Dense
from keras.layers import Flatten
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D
from keras.optimizers import SGD
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.layers import Dropout
from keras.layers.normalization import BatchNormalization
```

Image Preprocessing

In [ ]:
```
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the fi
traindir = "../input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/
validdir = "../input/new-plant-diseases-dataset/new plant diseases dataset(augmented)/
testdir = "../input/new-plant-diseases-dataset/test/test"

train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   fill_mode='nearest')

valid_datagen = ImageDataGenerator(rescale=1./255)

batch_size = 128
training_set = train_datagen.flow_from_directory(traindir,
                                                 target_size=(224, 224),
                                                 batch_size=batch_size,
                                                 class_mode='categorical')

valid_set = valid_datagen.flow_from_directory(validdir,
                                              target_size=(224, 224),
                                              batch_size=batch_size,
                                              class_mode='categorical')
```

In [ ]:
```
class_dict = training_set.class_indices
print(class_dict)
```

In [ ]:
```
li = list(class_dict.keys())
print(li)
```

In [ ]:
```
train_num = training_set.samples
valid_num = valid_set.samples
```

Model Preparation

In [26]:
```
base_model=VGG16(include_top=False,input_shape=(224,224,3))
base_model.trainable=False
```

In [27]:
```
classifier=keras.models.Sequential()
classifier.add(base_model)
classifier.add(Flatten())
classifier.add(Dense(38,activation='softmax'))
classifier.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_4 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_6 (Dense)              (None, 38)                953382
=================================================================
Total params: 15,668,070
Trainable params: 953,382
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [30]:
```
classifier.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [31]:
```
#fitting images to CNN
history = classifier.fit(training_set,
                         steps_per_epoch=train_num//batch_size,
                         validation_data=valid_set,
                         epochs=5,
                         validation_steps=valid_num//batch_size,
                         )
#saving model
#filepath="Mymodel.hdf5"
#model.save(filepath)
```

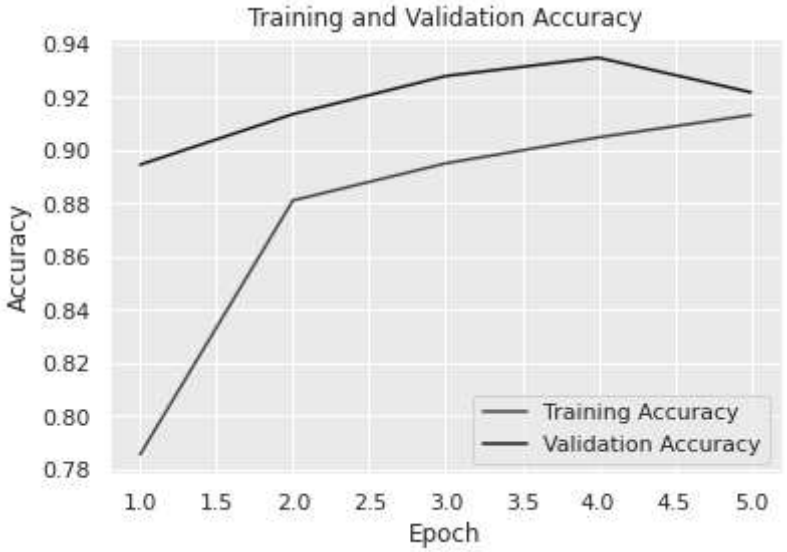
```
Epoch 1/5
 28/549 [>.............................] - ETA: 25:01 - loss: 2.7435 - accuracy: 0.30
83
```

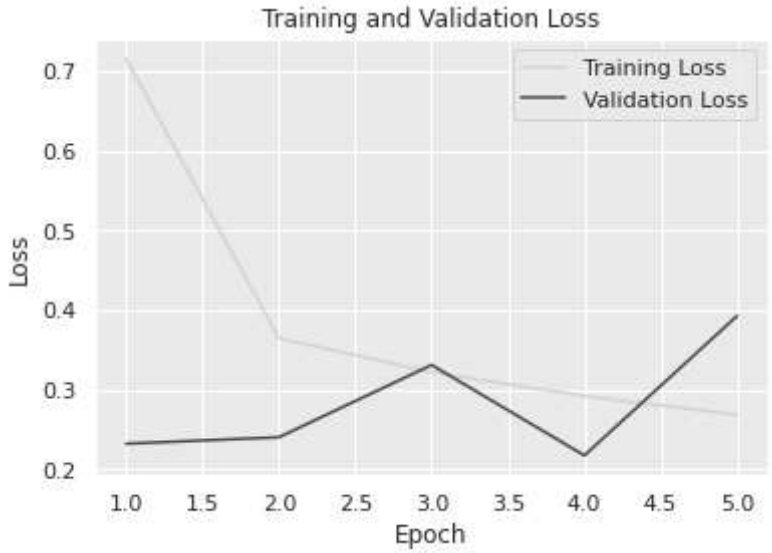
```
/opt/conda/lib/python3.7/site-packages/keras/utils/data_utils.py:616: UserWarning: Th
e input 207 could not be retrieved. It could be because a worker has died.
  UserWarning)
549/549 [==============================] - 1383s 3s/step - loss: 0.7155 - accuracy:
0.7857 - val_loss: 0.2321 - val_accuracy: 0.8946
Epoch 2/5
549/549 [==============================] - 1156s 2s/step - loss: 0.3646 - accuracy:
0.8812 - val_loss: 0.2402 - val_accuracy: 0.9136
Epoch 3/5
549/549 [==============================] - 1185s 2s/step - loss: 0.3222 - accuracy:
0.8951 - val_loss: 0.3311 - val_accuracy: 0.9279
Epoch 4/5
549/549 [==============================] - 1181s 2s/step - loss: 0.2916 - accuracy:
0.9049 - val_loss: 0.2173 - val_accuracy: 0.9348
Epoch 5/5
549/549 [==============================] - 1161s 2s/step - loss: 0.2690 - accuracy:
0.9133 - val_loss: 0.3927 - val_accuracy: 0.9218
```

In [32]:
```
#Saving our model
filepath="Mymodel.h5"
classifier.save(filepath)
```

Visualizing the Accuracy

In [34]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```





In [35]:
```
# predicting an image
from keras.preprocessing import image
import numpy as np
image_path = "../input/new-plant-diseases-dataset/test/test/TomatoEarlyBlight1.JPG"
new_img = image.load_img(image_path, target_size=(224, 224))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
img = img/255

print("Following is our prediction:")
prediction = classifier.predict(img)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
d = prediction.flatten()
j = d.max()
for index,item in enumerate(d):
    if item == j:
        class_name = li[index]

#ploting image with predicted class name
plt.figure(figsize = (4,4))
plt.imshow(new_img)
plt.axis('off')
```

```
plt.title(class_name)
plt.show()
```

Following is our prediction: