In [1]:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from PIL import Image
import os
```

In [2]:

```
!git clone https://github.com/harshad12patre/crop_yield_prediction.git
```

```
Cloning into 'crop_yield_prediction'...
remote: Enumerating objects: 19804, done.
remote: Counting objects: 100% (19804/19804), done.
remote: Compressing objects: 100% (19774/19774), done.
remote: Total 19804 (delta 53), reused 19773 (delta 26), pack-reused 0
Receiving objects: 100% (19804/19804), 338.72 MiB | 42.78 MiB/s, done.
Resolving deltas: 100% (53/53), done.
Checking out files: 100% (19708/19708), done.
```

In [3]:

```
pretrained_model =
tf.keras.applications.VGG16(include_top=True,weights='imagenet',input_shape=(224,
224,3))
pretrained_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 3s 0us/step
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
```

```
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0         
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544 
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312  
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000   
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

In [4]:

```
pretrained_model = 
tf.keras.applications.VGG16(include_top=False,weights='imagenet',input_shape=(224
,224,3))
pretrained_model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
```

```
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

In [5]:

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

In [9]:

```
last_layer = pretrained_model.get_layer("block5_pool")
output = last_layer.output
X = tf.keras.layers.Flatten()(output)
X = tf.keras.layers.Dense(1024,activation="relu")(X)
X = tf.keras.layers.Dense(512,activation="relu")(X)
X = tf.keras.layers.Dense(13,activation="softmax")(X)
model = tf.keras.models.Model(inputs=pretrained_model.input,outputs=X)
tf.keras.backend.clear_session()
model.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"
])
```

In [10]:

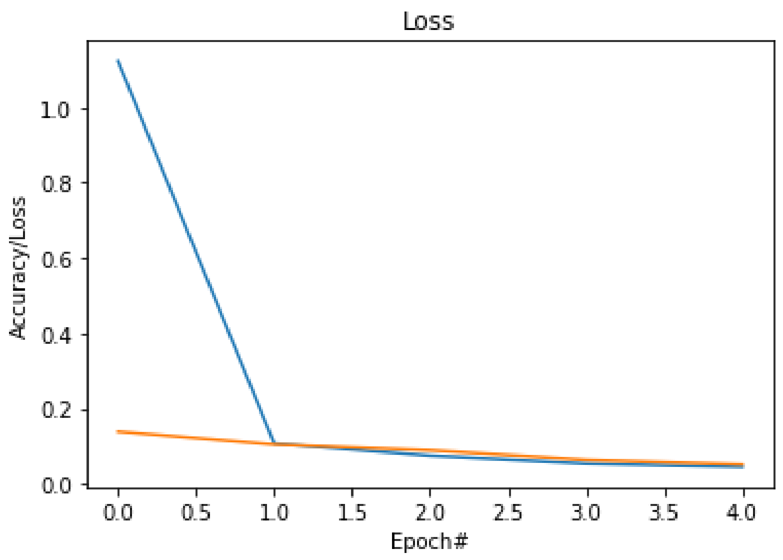
```
train_datagen = ImageDataGenerator(rescale=1/255.0, rotation_range=20,
      width_shift_range=0.1,
      height_shift_range=0.1,
      shear_range=0.1,
      zoom_range=0.1,
      horizontal_flip=True,
      fill_mode='nearest',
      validation_split=0.2)
batch_size = 128
train_generator =
train_datagen.flow_from_directory("/content/crop_yield_prediction/color"
                                                     ,batch_size=128,target_size=(2
,224),class_mode="categorical",subset="training")
validation_generator =
train_datagen.flow_from_directory("/content/crop_yield_prediction/color"
                                                     ,batch_size=128,target_size=(2
,224),class_mode="categorical",subset="validation")
history = model.fit(train_generator,steps_per_epoch=train_generator.samples//batc
h_size,validation_data=validation_generator,validation_steps=validation_generator
.samples//batch_size,epochs=5,verbose=1)
```

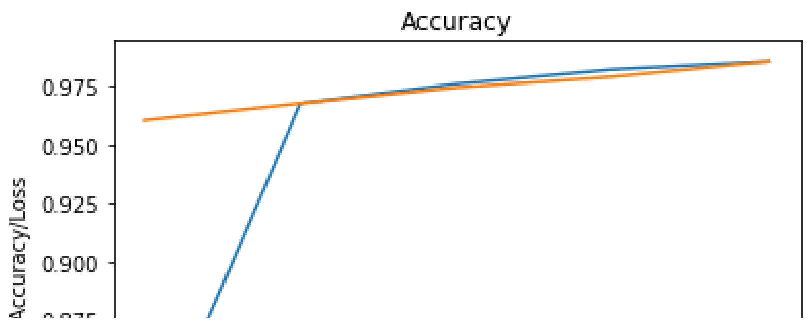
```
Found 15691 images belonging to 13 classes.
Found 3920 images belonging to 13 classes.
Epoch 1/5
```

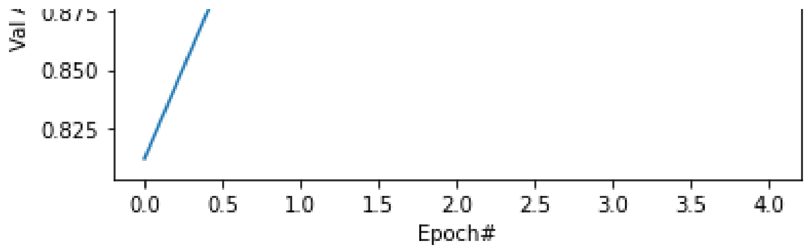
```
122/122 [==============================] - 230s 2s/step - loss: 2.8797 - accuracy
: 0.5964 - val_loss: 0.1390 - val_accuracy: 0.9604
Epoch 2/5
122/122 [==============================] - 226s 2s/step - loss: 0.1131 - accuracy
: 0.9680 - val_loss: 0.1055 - val_accuracy: 0.9677
Epoch 3/5
122/122 [==============================] - 226s 2s/step - loss: 0.0783 - accuracy
: 0.9747 - val_loss: 0.0899 - val_accuracy: 0.9745
Epoch 4/5
122/122 [==============================] - 227s 2s/step - loss: 0.0600 - accuracy
: 0.9804 - val_loss: 0.0646 - val_accuracy: 0.9792
Epoch 5/5
122/122 [==============================] - 226s 2s/step - loss: 0.0473 - accuracy
: 0.9858 - val_loss: 0.0527 - val_accuracy: 0.9857
```

In [11]:

```python
loss = history.history['loss']
accuracy = history.history['accuracy']
val_loss = history.history['val_loss']
val_accuracy = history.history['val_accuracy']
epoch = range(len(loss))
plt.figure()
plt.plot(epoch,loss)
plt.plot(epoch,val_loss)
plt.xlabel("Epoch#")
plt.ylabel("Accuracy/Loss")
plt.title("Loss")
plt.show()
plt.figure()
plt.plot(epoch,accuracy)
plt.plot(epoch,val_accuracy)
plt.xlabel("Epoch#")
plt.ylabel("Val Accuracy/Loss")
plt.title("Accuracy")
plt.show()
```






Val A
0.875
0.850
0.825
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epoch#