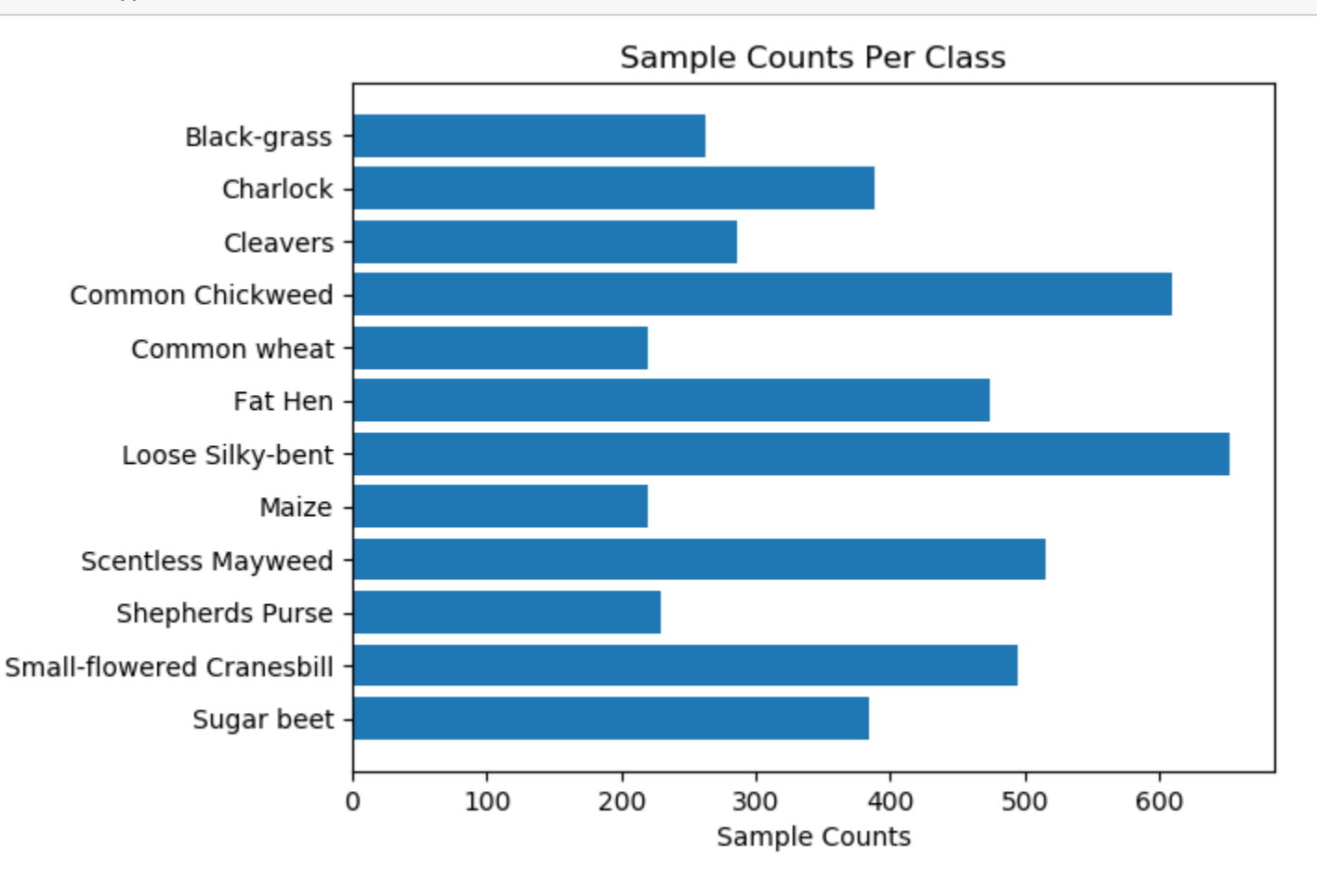
In [1]:
```python
#Import library
import pandas as pd
import numpy as np
import keras
import os
import matplotlib.pyplot as plt
from keras.preprocessing import image
import random
import pickle
from keras import models, layers, callbacks
from keras.callbacks import ModelCheckpoint
from keras.models import load_model
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import shutil
import cv2
from math import sqrt, floor
from prettytable import PrettyTable
```

In [2]:
```python
classes= []
sample_counts= []

for f in os.listdir(r'D:\Plant seed classification Model\train'):
    train_class_path= os.path.join(r'D:\Plant seed classification Model\train', f)
    if os.path.isdir(train_class_path):
        classes.append(f)
        sample_counts.append(len(os.listdir(train_class_path)))

plt.rcdefaults()
fig, ax = plt.subplots()

# Example data
y_pos = np.arange(len(classes))

ax.barh(y_pos, sample_counts, align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(classes)
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel('Sample Counts')
ax.set_title('Sample Counts Per Class')

plt.show()
```



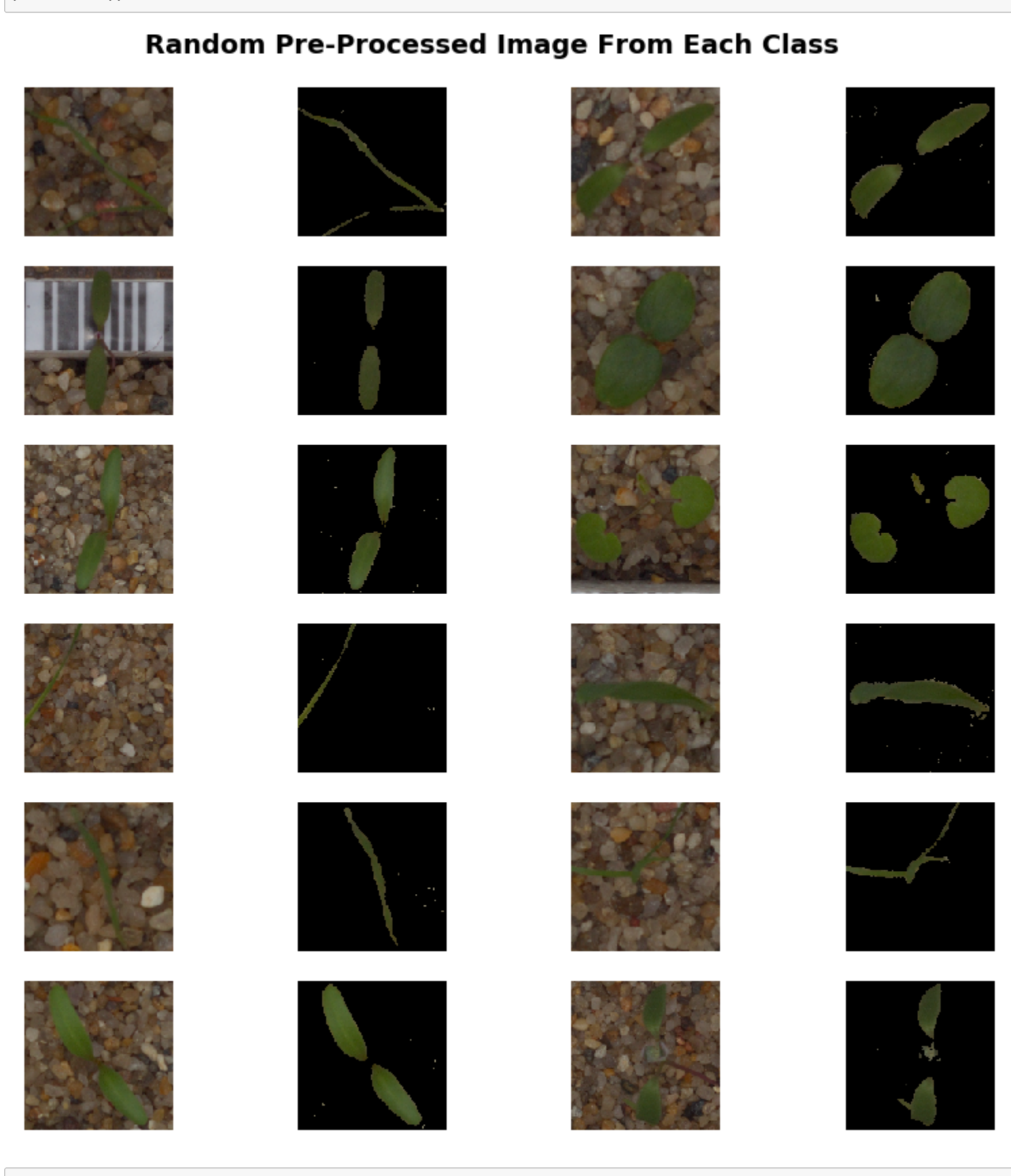
In [3]:
```python
#create validation set
def create_validation(validation_split=0.2):


    os.mkdir(r'D:\Plant seed classification Model\validation')
    for f in os.listdir(r'D:\Plant seed classification Model\train'):
        train_class_path= os.path.join(r'D:\Plant seed classification Model\train', f)
        if os.path.isdir(train_class_path):
            validation_class_path= os.path.join(r'D:\Plant seed classification Model\validat
ion', f)
            os.mkdir(validation_class_path)
            files_to_move= int(0.2*len(os.listdir(train_class_path)))
            random_image= os.path.join(train_class_path, random.choice(os.listdir(train_clas
s_path)))
            shutil.move(random_image, validation_class_path)
    print('Validation set created successfully using {:.2%} of training data'.format(validat
ion_split))
    return
```

In [4]:
```python
create_validation()
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
<ipython-input-4-a95990108f98> in <module>
----> 1 create_validation()

<ipython-input-3-8b196c0cfc84> in create_validation(validation_split)
      3
      4
----> 5     os.mkdir(r'D:\Plant seed classification Model\validation')
      6     for f in os.listdir(r'D:\Plant seed classification Model\train'):
      7         train_class_path= os.path.join(r'D:\Plant seed classification Model\train', f
)

FileExistsError: [WinError 183] Cannot create a file when that file already exists: 'D:\\Plan
t seed classification Model\\validation'
```

In [ ]:
```python
sample_counts= {}

for i, d in enumerate([r'D:\Plant seed classification Model\train', r'D:\Plant seed classifi
cation Model\validation']):

    classes= []
    sample_counts[d]= []

    for f in os.listdir(d):
        train_class_path= os.path.join(d, f)
        if os.path.isdir(train_class_path):
            classes.append(f)
            sample_counts[d].append(len(os.listdir(train_class_path)))

    #fig, ax= plt.subplot(221+i)
    fig, ax = plt.subplots()

    # Example data
    y_pos = np.arange(len(classes))

    ax.barh(y_pos, sample_counts[d], align='center')
    ax.set_yticks(y_pos)
    ax.set_yticklabels(classes)
    ax.invert_yaxis()  # labels read top-to-bottom
    ax.set_xlabel('Sample Counts')
    ax.set_title('{} Sample Counts Per Class'.format(d.capitalize()))

plt.show()
```

In [5]:
```python
lower_bound= (24, 50, 0)
upper_bound= (55, 255, 255)

fig= plt.figure(figsize=(10, 10))
fig.suptitle('Random Pre-Processed Image From Each Class', fontsize=14, y=.92, horizontalali
gnment='center', weight='bold')

for i in range(12):
    sample_class=os.path.join(r'D:\Plant seed classification Model\test')   #preprocessing o
f images of test dataset
    random_image= os.path.join(sample_class, random.choice(os.listdir(sample_class)))
    img= cv2.imread(random_image)
    img= cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img= cv2.resize(img, (150, 150))

    hsv_img= cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
    mask = cv2.inRange(hsv_img, lower_bound, upper_bound)
    result = cv2.bitwise_and(img, img, mask=mask)

    fig.add_subplot(6, 4, i*2+1)
    plt.imshow(img)
    plt.axis('off')

    fig.add_subplot(6, 4, i*2+2)
    plt.imshow(result)
    plt.axis('off')

plt.show()
```



In [6]:
```python
def color_segment_function(img_array):
    img_array= np.rint(img_array)
    img_array= img_array.astype('uint8')
    hsv_img= cv2.cvtColor(img_array, cv2.COLOR_RGB2HSV)
    mask = cv2.inRange(hsv_img, (24, 50, 0), (55, 255, 255))
    result = cv2.bitwise_and(img_array, img_array, mask=mask)
    result= result.astype('float64')
    return result

#image function from keras.preprocessing
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    brightness_range=[0.4,1],
    rescale=1.0/255.0)

test_datagen = image.ImageDataGenerator(rescale=1./255, preprocessing_function=color_segment
_function)
```

In [7]:
```python
#divide our train and test folder into training and testing dataset.
training_set = train_datagen.flow_from_directory(r'D:\Plant seed classification Model\train'
,
                                                 target_size = (150, 150),
                                                 batch_size = 20,
                                                 class_mode = 'categorical')
```

```
Found 4738 images belonging to 12 classes.
```

In [8]:
```python
test_set = test_datagen.flow_from_directory(r'D:\Plant seed classification Model\validation'
,
                                            target_size = (150, 150),
                                            batch_size = 20,
                                            class_mode = 'categorical')
```

```
Found 12 images belonging to 12 classes.
```

In [9]:
```python
# using neural network :
#Model Traning part begins
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), input_shape=(150, 150, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.1))


model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.1))



model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.1))


model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.1))


model.add(layers.Flatten())
model.add(layers.Dropout(0.4))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.4))

model.add(layers.Dense(12, activation='softmax'))
```

In [10]:
```python
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
dropout (Dropout)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 36, 36, 64)	0
dropout_1 (Dropout)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 17, 17, 128)	0
dropout_2 (Dropout)	(None, 17, 17, 128)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 7, 7, 128)	0
dropout_3 (Dropout)	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dropout_4 (Dropout)	(None, 6272)	0
dense (Dense)	(None, 256)	1605888
dropout_5 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 12)	3084

Total params: 1,849,804
Trainable params: 1,849,804
Non-trainable params: 0

In [11]:
```python
best_cb= callbacks.ModelCheckpoint('model_best.h5',
                                   monitor='val_loss',
                                   verbose=1,
                                   save_best_only=True,
                                   save_weights_only=False,
                                   mode='auto',
                                   period=1)

opt= keras.optimizers.Adam(lr=0.0005, amsgrad=True)   # Learning rate=0.0005
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify the fre
quency in number of batches seen.
C:\Users\Priya Mittal\Anaconda3\lib\site-packages\tensorflow\python\keras\optimizer_v2\optimi
zer_v2.py:375: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
```

In [12]:
```python
model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=30,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set),
  callbacks= [best_cb])
```

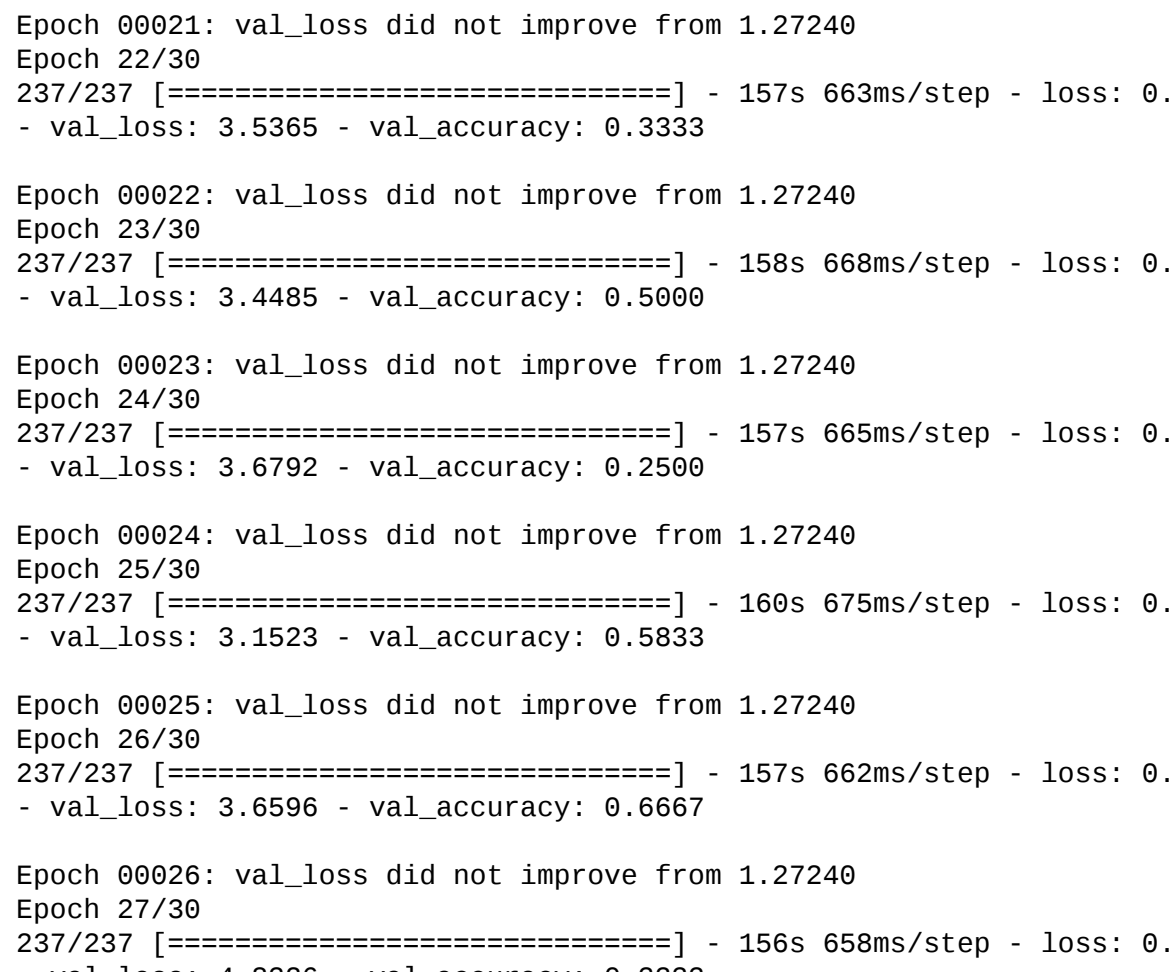
```
C:\Users\Priya Mittal\Anaconda3\lib\site-packages\keras\engine\training.py:1915: UserWarning:
`Model.fit_generator` is deprecated and will be removed in a future version. Please use `Mode
l.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/30
237/237 [==============================] - 441s 2s/step - loss: 2.3847 - accuracy: 0.1557 - v
al_loss: 3.4488 - val_accuracy: 0.0833

Epoch 00001: val_loss improved from inf to 3.44880, saving model to model_best.h5
Epoch 2/30
237/237 [==============================] - 166s 700ms/step - loss: 1.8904 - accuracy: 0.3304
- val_loss: 3.0187 - val_accuracy: 0.1667

Epoch 00002: val_loss improved from 3.44880 to 3.01867, saving model to model_best.h5
Epoch 3/30
237/237 [==============================] - 157s 662ms/step - loss: 1.6130 - accuracy: 0.4277
- val_loss: 2.5012 - val_accuracy: 0.0833

Epoch 00003: val_loss improved from 3.01867 to 2.50116, saving model to model_best.h5
Epoch 4/30
237/237 [==============================] - 157s 664ms/step - loss: 1.4590 - accuracy: 0.4996
- val_loss: 3.0122 - val_accuracy: 0.1667

Epoch 00004: val_loss did not improve from 2.50116
Epoch 5/30
237/237 [==============================] - 154s 651ms/step - loss: 1.3768 - accuracy: 0.5345
- val_loss: 3.2960 - val_accuracy: 0.1667

Epoch 00005: val_loss did not improve from 2.50116
Epoch 6/30
237/237 [==============================] - 154s 652ms/step - loss: 1.1751 - accuracy: 0.5937
- val_loss: 4.2296 - val_accuracy: 0.1667

Epoch 00006: val_loss did not improve from 2.50116
Epoch 7/30
237/237 [==============================] - 155s 655ms/step - loss: 1.1071 - accuracy: 0.6240
- val_loss: 4.1459 - val_accuracy: 0.1667

Epoch 00007: val_loss did not improve from 2.50116
Epoch 8/30
237/237 [==============================] - 1095s 5s/step - loss: 0.9455 - accuracy: 0.6716 -
val_loss: 4.4960 - val_accuracy: 0.1667

Epoch 00008: val_loss did not improve from 2.50116
Epoch 9/30
237/237 [==============================] - 219s 926ms/step - loss: 0.9313 - accuracy: 0.6792
- val_loss: 4.0388 - val_accuracy: 0.1667

Epoch 00009: val_loss did not improve from 2.50116
Epoch 10/30
237/237 [==============================] - 268s 1s/step - loss: 0.8332 - accuracy: 0.7066 - v
al_loss: 2.6689 - val_accuracy: 0.3333

Epoch 00010: val_loss did not improve from 2.50116
Epoch 11/30
237/237 [==============================] - 1106s 5s/step - loss: 0.8414 - accuracy: 0.7192 -
val_loss: 5.1599 - val_accuracy: 0.1667

Epoch 00011: val_loss did not improve from 2.50116
Epoch 12/30
237/237 [==============================] - 271s 1s/step - loss: 0.7419 - accuracy: 0.7363 - v
al_loss: 4.7227 - val_accuracy: 0.1667

Epoch 00012: val_loss did not improve from 2.50116
Epoch 13/30
237/237 [==============================] - 270s 1s/step - loss: 0.7229 - accuracy: 0.7564 - v
al_loss: 2.7886 - val_accuracy: 0.3333

Epoch 00013: val_loss did not improve from 2.50116
Epoch 14/30
237/237 [==============================] - 4298s 18s/step - loss: 0.6963 - accuracy: 0.7616 -
val_loss: 4.6501 - val_accuracy: 0.2500

Epoch 00014: val_loss did not improve from 2.50116
Epoch 15/30
237/237 [==============================] - 205s 865ms/step - loss: 0.6405 - accuracy: 0.7741
- val_loss: 2.4173 - val_accuracy: 0.3333

Epoch 00015: val_loss improved from 2.50116 to 2.41730, saving model to model_best.h5
Epoch 16/30
237/237 [==============================] - 155s 655ms/step - loss: 0.6426 - accuracy: 0.7757
- val_loss: 1.2724 - val_accuracy: 0.5833

Epoch 00016: val_loss improved from 2.41730 to 1.27240, saving model to model_best.h5
Epoch 17/30
237/237 [==============================] - 156s 658ms/step - loss: 0.5973 - accuracy: 0.7925
- val_loss: 1.8575 - val_accuracy: 0.5000

Epoch 00017: val_loss did not improve from 1.27240
Epoch 18/30
237/237 [==============================] - 155s 655ms/step - loss: 0.6125 - accuracy: 0.7782
- val_loss: 1.8742 - val_accuracy: 0.4167

Epoch 00018: val_loss did not improve from 1.27240
Epoch 19/30
237/237 [==============================] - 157s 665ms/step - loss: 0.5578 - accuracy: 0.8036
- val_loss: 1.7708 - val_accuracy: 0.6667

Epoch 00019: val_loss did not improve from 1.27240
Epoch 20/30
237/237 [==============================] - 158s 666ms/step - loss: 0.5897 - accuracy: 0.7981
- val_loss: 2.5739 - val_accuracy: 0.6667

Epoch 00020: val_loss did not improve from 1.27240
Epoch 21/30
237/237 [==============================] - 163s 687ms/step - loss: 0.5818 - accuracy: 0.7987
- val_loss: 5.4892 - val_accuracy: 0.1667

Epoch 00021: val_loss did not improve from 1.27240
Epoch 22/30
237/237 [==============================] - 157s 663ms/step - loss: 0.5395 - accuracy: 0.8091
- val_loss: 3.5365 - val_accuracy: 0.3333

Epoch 00022: val_loss did not improve from 1.27240
Epoch 23/30
237/237 [==============================] - 158s 668ms/step - loss: 0.5416 - accuracy: 0.8121
- val_loss: 3.4485 - val_accuracy: 0.5000

Epoch 00023: val_loss did not improve from 1.27240
Epoch 24/30
237/237 [==============================] - 157s 665ms/step - loss: 0.4935 - accuracy: 0.8242
- val_loss: 3.6792 - val_accuracy: 0.2500

Epoch 00024: val_loss did not improve from 1.27240
Epoch 25/30
237/237 [==============================] - 160s 675ms/step - loss: 0.5114 - accuracy: 0.8161
- val_loss: 3.1523 - val_accuracy: 0.5833

Epoch 00025: val_loss did not improve from 1.27240
Epoch 26/30
237/237 [==============================] - 157s 662ms/step - loss: 0.5121 - accuracy: 0.8131
- val_loss: 3.6596 - val_accuracy: 0.6667

Epoch 00026: val_loss did not improve from 1.27240
Epoch 27/30
237/237 [==============================] - 156s 658ms/step - loss: 0.4735 - accuracy: 0.8294
- val_loss: 4.8326 - val_accuracy: 0.3333

Epoch 00027: val_loss did not improve from 1.27240
Epoch 28/30
237/237 [==============================] - 155s 654ms/step - loss: 0.4892 - accuracy: 0.8330
- val_loss: 2.8849 - val_accuracy: 0.5833

Epoch 00028: val_loss did not improve from 1.27240
Epoch 29/30
237/237 [==============================] - 159s 672ms/step - loss: 0.4887 - accuracy: 0.8380
- val_loss: 6.9572 - val_accuracy: 0.1667

Epoch 00029: val_loss did not improve from 1.27240
Epoch 30/30
237/237 [==============================] - 157s 662ms/step - loss: 0.4703 - accuracy: 0.8359
- val_loss: 6.5402 - val_accuracy: 0.1667

Epoch 00030: val_loss did not improve from 1.27240
```

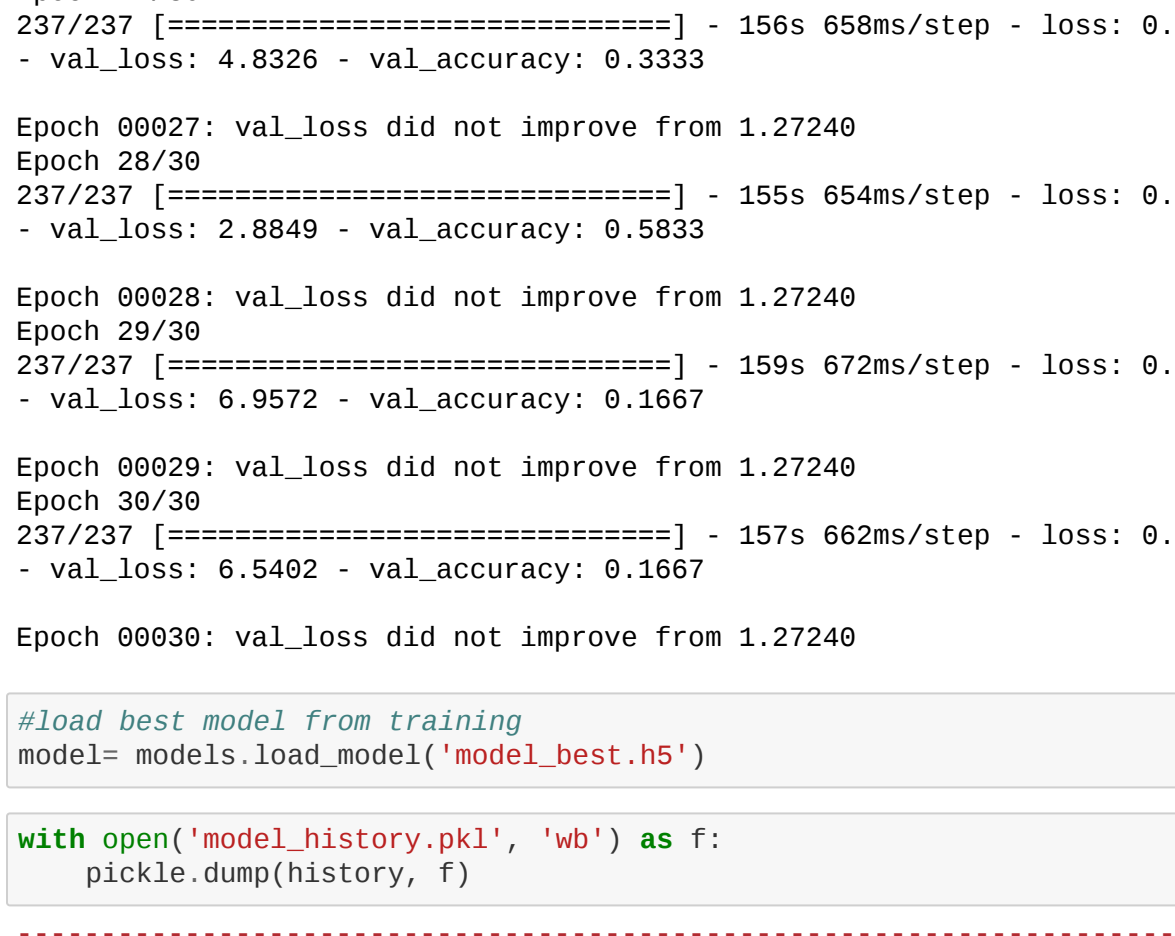
In [13]:
```python
#load best model from training
model= models.load_model('model_best.h5')
```

In [14]:
```python
with open('model_history.pkl', 'wb') as f:
    pickle.dump(history, f)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-55dcc661aa88> in <module>
      1 with open('model_history.pkl', 'wb') as f:
----> 2     pickle.dump(history, f)

TypeError: can't pickle weakref objects
```

In [16]:
```python
# plot the loss
import matplotlib.pyplot as plt
plt.plot(history.history['loss'], label='train loss')
#plt.plot(history.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(history.history['accuracy'], label='train acc')
#plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```






```
<Figure size 640x480 with 0 Axes>
```

In [18]:
```python
pred= model.predict_generator(test_set, steps= test_set.n, verbose=1)
```

```
12/12 [==============================] - 1s 109ms/step
```

In [21]:
```python
from utils import label_map_util
predicted_class_indices=np.argmax(pred,axis=1)

prediction_labels = [label_map_util[k] for k in predicted_class_indices]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-3e290abd08ca> in <module>
      2 predicted_class_indices=np.argmax(pred,axis=1)
      3
----> 4 prediction_labels = [label_map_util[k] for k in predicted_class_indices]

<ipython-input-21-3e290abd08ca> in <listcomp>(.0)
      2 predicted_class_indices=np.argmax(pred,axis=1)
      3
----> 4 prediction_labels = [label_map_util[k] for k in predicted_class_indices]

NameError: name 'label_map_util' is not defined
```

In [22]:
```python
filenames= test_set.filenames
```

In [24]:
```python
#Final Result time for predicting the soln for some images dataset To check our model valid
 or not for predicting the results.
import csv
csvfile= open(r'D:\Plant seed classification Model\result_sample', 'w', newline='')
writer= csv.writer(csvfile)
headers= ['file', 'species']

writer.writerow(headers)
t = PrettyTable(headers)
for i, f, p in zip(range(len(filenames)), filenames, prediction_labels):
    writer.writerow([os.path.basename(f),p])
    if i <10:
        t.add_row([os.path.basename(f), p])
    elif i<13:
        t.add_row(['.', '.'])
csvfile.close()
print(t)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-cda96c2a860c> in <module>
      7 writer.writerow(headers)
      8 t = PrettyTable(headers)
----> 9 for i, f, p in zip(range(len(filenames)), filenames, prediction_labels):
     10     writer.writerow([os.path.basename(f),p])
     11     if i <10:

NameError: name 'prediction_labels' is not defined
```

In [ ]: