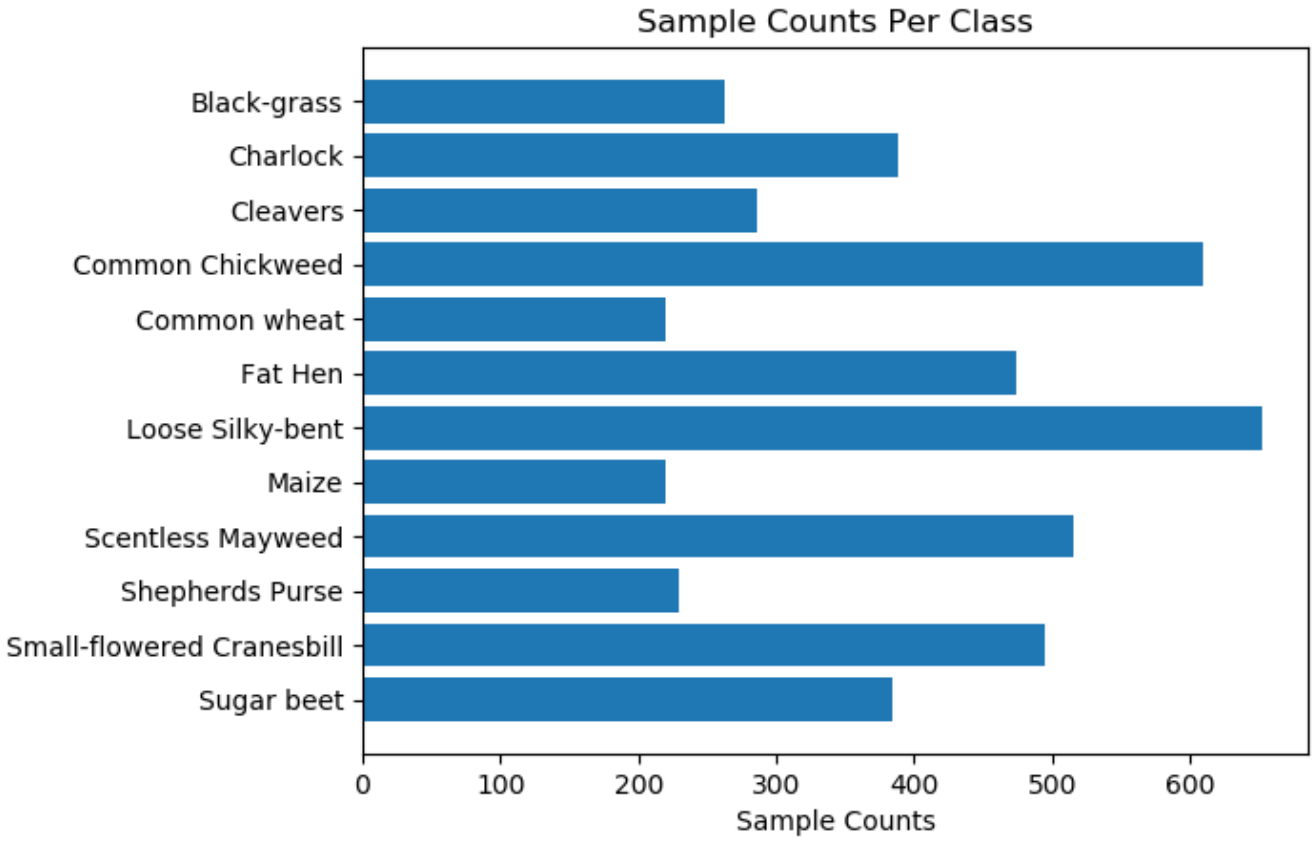
In [2]:

```
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 [3]:

```
classes= []
sample_counts= []

for f in os.listdir(r'D:\seed Classification  Model\train'):
    train_class_path= os.path.join(r'D:\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()
```



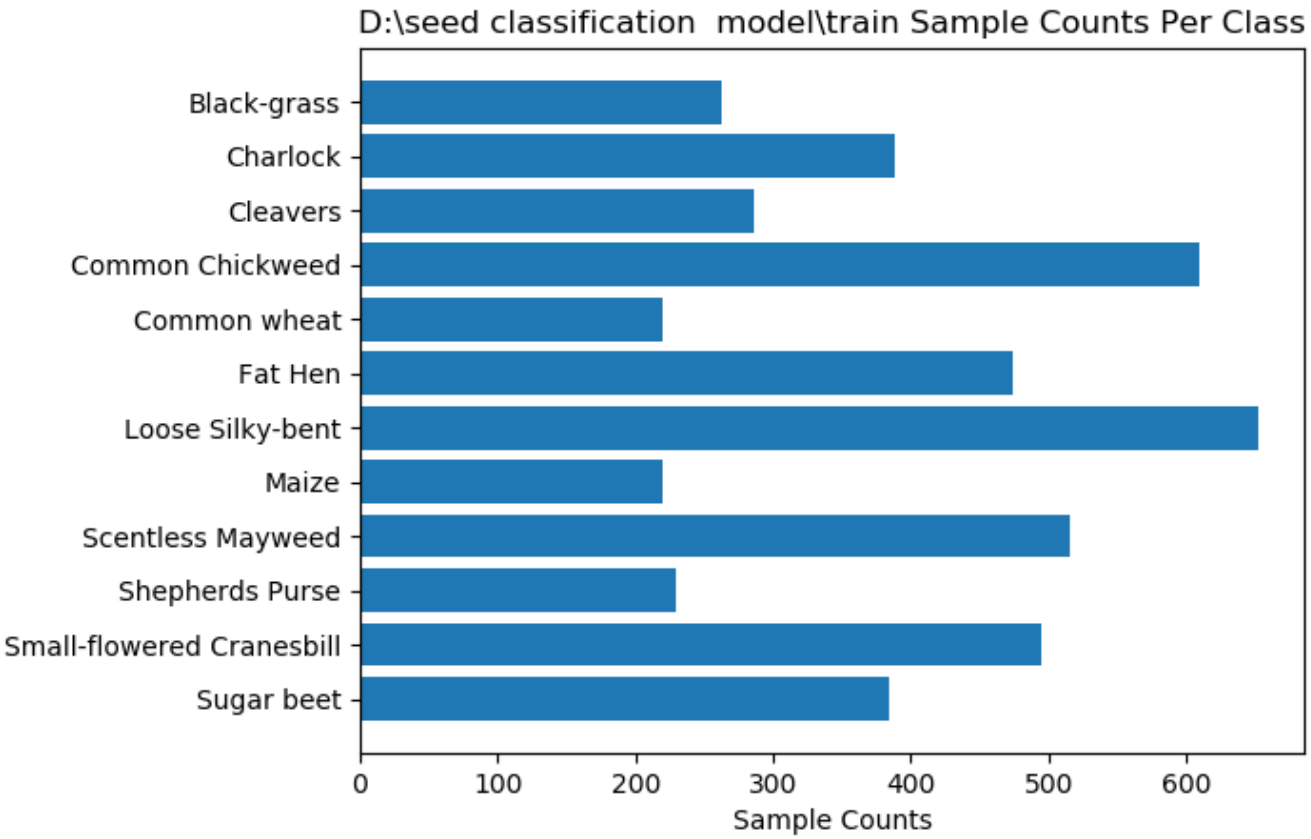
In [4]:
```
#create validation set
def create_validation(validation_split=0.2):
    if os.path.isdir('validation'):
        print('Validation directory already created!')
        print('Process Terminated')
        return

    os.mkdir(r'D:\seed Classification  Model\validation')
    for f in os.listdir(r'D:\seed Classification  Model\train'):
        train_class_path= os.path.join(r'D:\seed Classification  Model\train',
f)
        if os.path.isdir(train_class_path):
            validation_class_path= os.path.join(r'D:\seed Classification  Model
\validation', 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.listd
ir(train_class_path)))
            shutil.move(random_image, validation_class_path)
    print('Validation set created successfully using {:.2%} of training data'.f
ormat(validation_split))
```

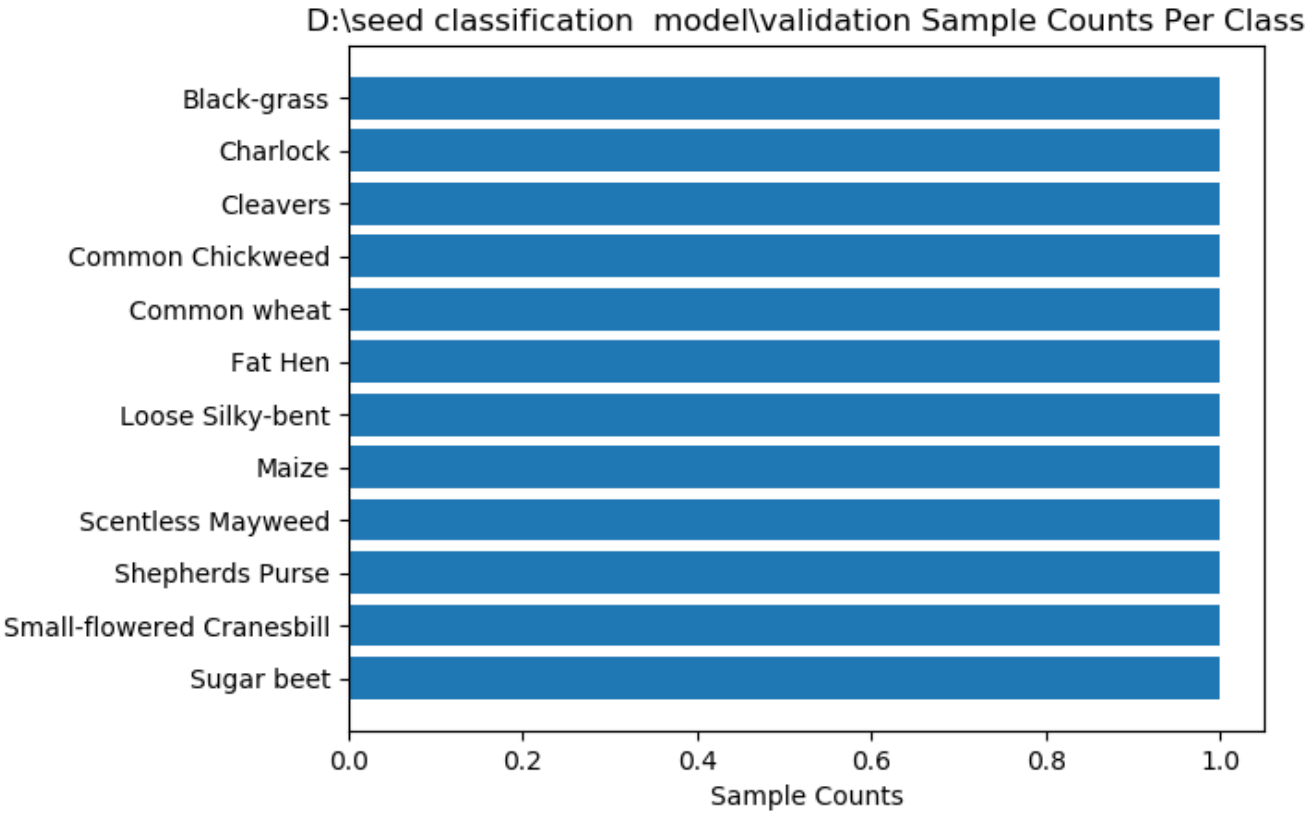
In [5]:
```
create_validation()
```

```
Validation directory already created!
Process Terminated
```

In [6]:

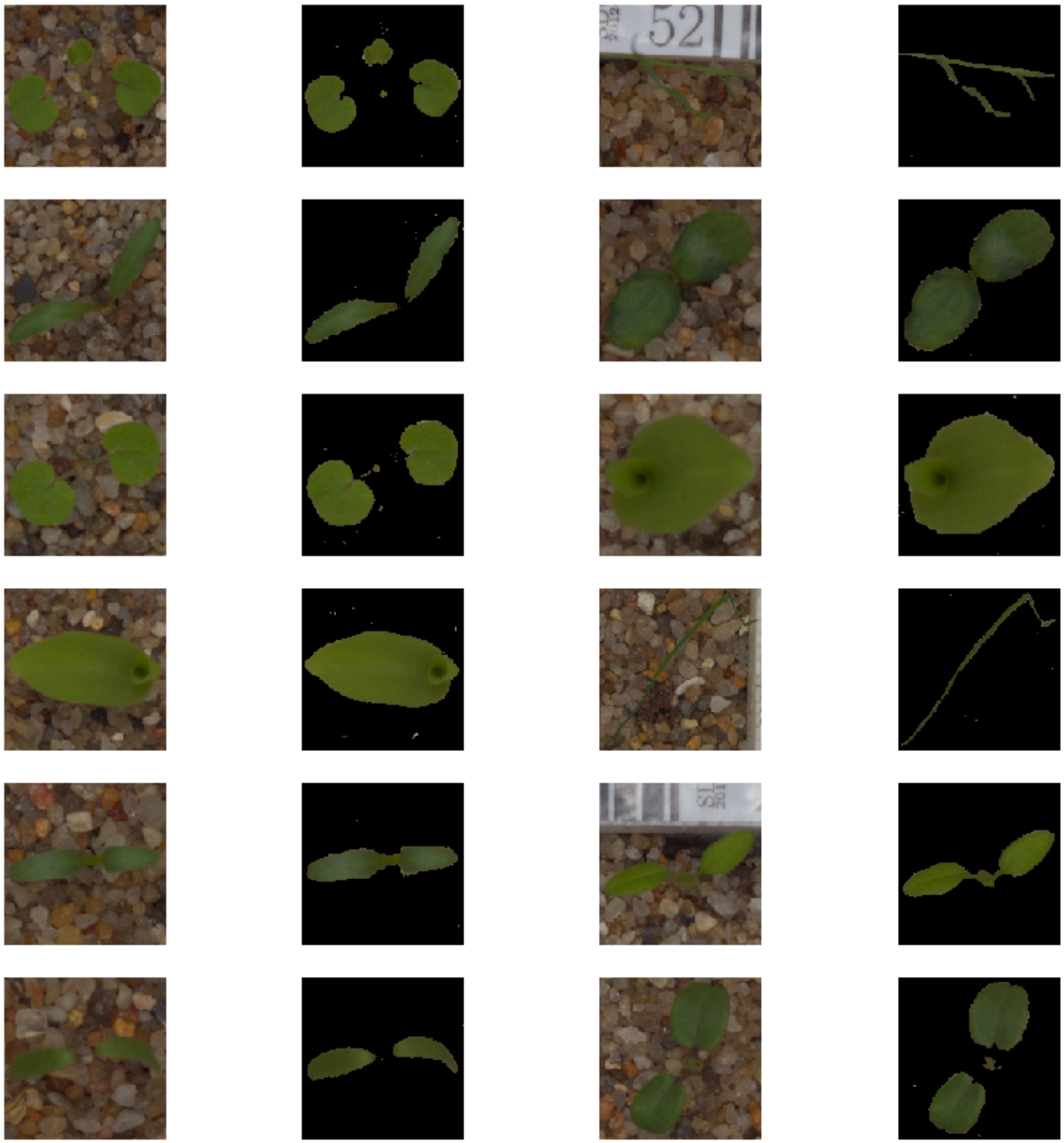
```
sample_counts= {}

for i, d in enumerate([r'D:\seed Classification  Model\train', r'D:\seed Classi
fication  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()
```


D:\seed classification model\train Sample Counts Per Class
Black-grass
Charlock
Cleavers
Common Chickweed
Common wheat
Fat Hen
Loose Silky-bent
Maize
Scentless Mayweed
Shepherds Purse
Small-flowered Cranesbill
Sugar beet
0
100
200
300
400
500
600
Sample Counts


D:\seed classification model\validation Sample Counts Per Class
Black-grass
Charlock
Cleavers
Common Chickweed
Common wheat
Fat Hen
Loose Silky-bent
Maize
Scentless Mayweed
Shepherds Purse
Small-flowered Cranesbill
Sugar beet
0.0
0.2
0.4
0.6
0.8
1.0
Sample Counts

In [7]:

```
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,
horizontalalignment='center', weight='bold')

for i in range(12):
    sample_class=os.path.join(r'D:\seed Classification  Model\test')
    random_image= os.path.join(sample_class, random.choice(os.listdir(sample_cl
ass)))
    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 [8]:
```
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
```

In [9]:
```
#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 [10]:
```
training_set = train_datagen.flow_from_directory(r'D:\seed Classification  Mode
l\train',
                                                 target_size = (150, 150),
                                                 batch_size = 20,
                                                 class_mode = 'categorical')
```

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

In [50]:
```
test_set = test_datagen.flow_from_directory(r'D:\seed Classification  Model\val
idation',
                                            target_size = (150, 150),
                                            batch_size = 20,
                                            class_mode = 'categorical')
```

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

In [12]:

```
#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'))
```

```
WARNING:tensorflow:From C:\Users\Parth Madan\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [13]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout_4 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_5 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               1605888
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 12)                3084
=================================================================
Total params: 1,849,804
Trainable params: 1,849,804
Non-trainable params: 0
_________________________________________________________________
```

In [14]:
```
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=0.00005
```

In [15]:

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

history = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=50,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set),
  callbacks= [best_cb])
```

```
WARNING:tensorflow:From C:\Users\Parth Madan\Anaconda3\lib\site-packages\keras
\backend\tensorflow_backend.py:422: The name tf.global_variables is deprecated.
Please use tf.compat.v1.global_variables instead.

Epoch 1/50
237/237 [==============================] - 334s 1s/step - loss: 2.2886 - accura
cy: 0.2018
Epoch 2/50

C:\Users\Parth Madan\Anaconda3\lib\site-packages\keras\callbacks\callbacks.py:7
07: RuntimeWarning: Can save best model only with val_loss available, skipping.
  'skipping.' % (self.monitor), RuntimeWarning)
```

```
237/237 [==============================] - 242s 1s/step - loss: 1.8447 - accura
cy: 0.3527
Epoch 3/50
237/237 [==============================] - 243s 1s/step - loss: 1.5856 - accura
cy: 0.4491
Epoch 4/50
237/237 [==============================] - 385s 2s/step - loss: 1.4449 - accura
cy: 0.5065
Epoch 5/50
237/237 [==============================] - 347s 1s/step - loss: 1.2917 - accura
cy: 0.5585
Epoch 6/50
237/237 [==============================] - 260s 1s/step - loss: 1.2246 - accura
cy: 0.5678
Epoch 7/50
237/237 [==============================] - 231s 973ms/step - loss: 1.1328 - acc
uracy: 0.6173
Epoch 8/50
237/237 [==============================] - 233s 982ms/step - loss: 1.0363 - acc
uracy: 0.6522
Epoch 9/50
237/237 [==============================] - 230s 972ms/step - loss: 0.9924 - acc
uracy: 0.6638
Epoch 10/50
237/237 [==============================] - 241s 1s/step - loss: 0.9020 - accura
cy: 0.6933
Epoch 11/50
237/237 [==============================] - 241s 1s/step - loss: 0.8682 - accura
cy: 0.7075
Epoch 12/50
237/237 [==============================] - 248s 1s/step - loss: 0.8252 - accura
cy: 0.7227
Epoch 13/50
237/237 [==============================] - 245s 1s/step - loss: 0.7613 - accura
cy: 0.7419
Epoch 14/50
237/237 [==============================] - 230s 973ms/step - loss: 0.7478 - acc
uracy: 0.7463
Epoch 15/50
237/237 [==============================] - 232s 978ms/step - loss: 0.7096 - acc
uracy: 0.7539
Epoch 16/50
237/237 [==============================] - 237s 998ms/step - loss: 0.6486 - acc
uracy: 0.7712
Epoch 17/50
237/237 [==============================] - 245s 1s/step - loss: 0.6759 - accura
cy: 0.7706
Epoch 18/50
237/237 [==============================] - 232s 981ms/step - loss: 0.6352 - acc
uracy: 0.7860
Epoch 19/50
237/237 [==============================] - 231s 976ms/step - loss: 0.5905 - acc
uracy: 0.7940
Epoch 20/50
237/237 [==============================] - 231s 973ms/step - loss: 0.5658 - acc
uracy: 0.8022
Epoch 21/50
237/237 [==============================] - 230s 970ms/step - loss: 0.5790 - acc
uracy: 0.8008
Epoch 22/50
237/237 [==============================] - 246s 1s/step - loss: 0.5469 - accura
```

```
cy: 0.8128
Epoch 23/50
237/237 [==============================] - 230s 971ms/step - loss: 0.5344 - acc
uracy: 0.8134
Epoch 24/50
237/237 [==============================] - 229s 966ms/step - loss: 0.5186 - acc
uracy: 0.8179
Epoch 25/50
237/237 [==============================] - 327s 1s/step - loss: 0.5019 - accura
cy: 0.8219
Epoch 26/50
237/237 [==============================] - 243s 1s/step - loss: 0.4969 - accura
cy: 0.8246
Epoch 27/50
237/237 [==============================] - 270s 1s/step - loss: 0.4824 - accura
cy: 0.8316
Epoch 28/50
237/237 [==============================] - 257s 1s/step - loss: 0.4507 - accura
cy: 0.8404
Epoch 29/50
237/237 [==============================] - 270s 1s/step - loss: 0.4535 - accura
cy: 0.8438
Epoch 30/50
237/237 [==============================] - 248s 1s/step - loss: 0.4600 - accura
cy: 0.8371
Epoch 31/50
237/237 [==============================] - 279s 1s/step - loss: 0.4598 - accura
cy: 0.8364
Epoch 32/50
237/237 [==============================] - 304s 1s/step - loss: 0.4414 - accura
cy: 0.8407
Epoch 33/50
237/237 [==============================] - 257s 1s/step - loss: 0.4192 - accura
cy: 0.8607
Epoch 34/50
237/237 [==============================] - 249s 1s/step - loss: 0.4187 - accura
cy: 0.8520
Epoch 35/50
237/237 [==============================] - 278s 1s/step - loss: 0.3709 - accura
cy: 0.8645
Epoch 36/50
237/237 [==============================] - 499s 2s/step - loss: 0.3842 - accura
cy: 0.8594
Epoch 37/50
237/237 [==============================] - 401s 2s/step - loss: 0.4021 - accura
cy: 0.8573
Epoch 38/50
237/237 [==============================] - 360s 2s/step - loss: 0.3777 - accura
cy: 0.8628
Epoch 39/50
237/237 [==============================] - 419s 2s/step - loss: 0.3738 - accura
cy: 0.8691
Epoch 40/50
237/237 [==============================] - 401s 2s/step - loss: 0.3775 - accura
cy: 0.8628
Epoch 41/50
237/237 [==============================] - 597s 3s/step - loss: 0.3747 - accura
cy: 0.8687
Epoch 42/50
237/237 [==============================] - 360s 2s/step - loss: 0.3552 - accura
cy: 0.8725
Epoch 43/50
```

```
237/237 [==============================] - 306s 1s/step - loss: 0.3566 - accura
cy: 0.8736
Epoch 44/50
237/237 [==============================] - 325s 1s/step - loss: 0.3619 - accura
cy: 0.8681
Epoch 45/50
237/237 [==============================] - 302s 1s/step - loss: 0.3609 - accura
cy: 0.8681
Epoch 46/50
237/237 [==============================] - 257s 1s/step - loss: 0.3539 - accura
cy: 0.8721
Epoch 47/50
237/237 [==============================] - 242s 1s/step - loss: 0.3397 - accura
cy: 0.8808
Epoch 48/50
237/237 [==============================] - 282s 1s/step - loss: 0.3279 - accura
cy: 0.8837
Epoch 49/50
237/237 [==============================] - 252s 1s/step - loss: 0.3392 - accura
cy: 0.8797
Epoch 50/50
237/237 [==============================] - 299s 1s/step - loss: 0.3230 - accura
cy: 0.8850
```

In [27]:
```
from tensorflow.keras.models import load_model
model.save('model_best.h5')
```

In [ ]:
```
```

In [28]:
```
#load best model from training
model= models.load_model('model_best.h5')
```

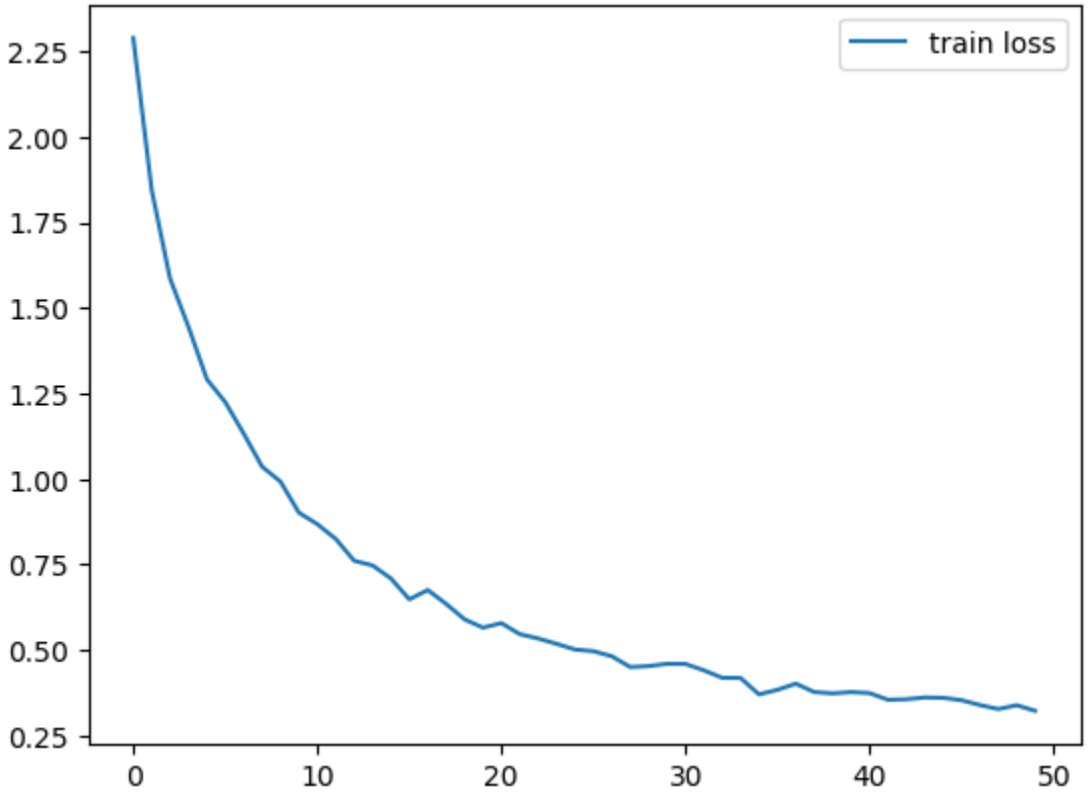
In [51]:
```
model
```

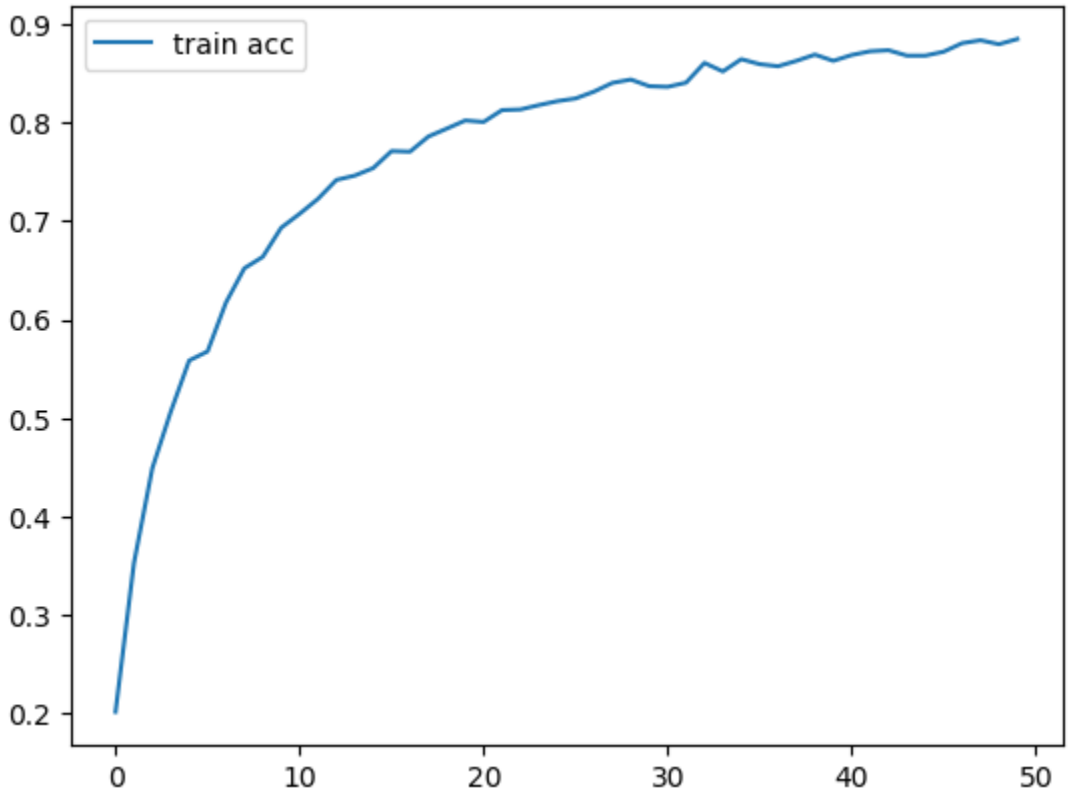
Out[51]:
```
<keras.engine.sequential.Sequential at 0x191801a8128>
```

In [29]:
```
#save history
with open('model_history.pkl', 'wb') as f:
    pickle.dump(history, f)
```

In [46]:

```
# 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 [53]:
```
pred= model.predict_generator(test_set, steps= test_set.n, verbose=1)
```

```
12/12 [==============================] - 7s 610ms/step
```

In [70]:
```
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]
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-70-0cf1c08cab7b> in <module>
----> 1 from utils import label_map_util
      2 predicted_class_indices=np.argmax(pred,axis=1)
      3
      4 prediction_labels = [label_map_util[k] for k in predicted_class_indices
]

ImportError: cannot import name 'label_map_util' from 'utils' (C:\Users\Parth M
adan\Anaconda3\lib\site-packages\utils\__init__.py)
```

In [63]:
```
filenames= test_set.filenames
```

In [64]:
```
#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:\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-64-10aa769677c6> 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 [42]:
```

```

Out[42]:
```
[]
```

In [ ]:
```

```