

```
In [1]: import numpy as np #dealing with arrays as model requires array to be passed
import os #read or write a file within directory
import cv2 #dealing with images extracting data from images
import pandas as pd #data manipulation and analysis
from tqdm import tqdm # for well-established ProgressBar
from random import shuffle #only shuffles the array along the first axis of a mul
LR = 1e-3
MODEL_NAME = 'plantclassification-{}-{}.model'.format(LR, '2conv-basic') # just so
```

```
In [2]: data_dir = ''
train_dir = os.path.join(data_dir, 'train')
test_dir = os.path.join(data_dir, 'test')
IMG_SIZE = 128
```

```
In [3]: #list of categories in array format
CATEGORIES = ['Black-grass', 'Charlock', 'Cleavers', 'Common Chickweed', 'Common
            'Maize', 'Scentless Mayweed', 'Shepherds Purse', 'Small-flowered Cr
NUM_CATEGORIES = len(CATEGORIES)
print (NUM_CATEGORIES)
```

12

```
In [4]: ''' function that accept plant category and return array format of the vlaue , or
        am sure there's better way to do this .....'''

def label_img(word_label):
    if word_label == 'Black-grass': return [1,0,0,0,0,0,0,0,0,0,0,0]
    elif word_label == 'Charlock': return [0,1,0,0,0,0,0,0,0,0,0,0]
    elif word_label == 'Cleavers': return [0,0,1,0,0,0,0,0,0,0,0,0]
    elif word_label == 'Common Chickweed': return [0,0,0,1,0,0,0,0,0,0,0,0]
    elif word_label == 'Common wheat': return [0,0,0,0,1,0,0,0,0,0,0,0]
    elif word_label == 'Fat Hen': return [0,0,0,0,0,1,0,0,0,0,0,0]
    elif word_label == 'Loose Silky-bent': return [0,0,0,0,0,0,1,0,0,0,0,0]
    elif word_label == 'Maize': return [0,0,0,0,0,0,0,1,0,0,0,0]
    elif word_label == 'Scentless Mayweed': return [0,0,0,0,0,0,0,0,1,0,0,0]
    elif word_label == 'Shepherds Purse': return [0,0,0,0,0,0,0,0,0,1,0,0]
    elif word_label == 'Small-flowered Cranesbill': return [0,0,0,0,0,0,0,0,0,0,1,0]
    elif word_label == 'Sugar beet': return [0,0,0,0,0,0,0,0,0,0,0,1]
```

In [5]:

```
'''function that will create train data , will go thought all the file do this
----read the image in  grayscale mode ,resize it
---change it to numpy arrays and  append it to dataframe train with it`s associat

def create_train_data():
    train = []
    for category_id, category in enumerate(CATEGORIES):
        for img in tqdm(os.listdir(os.path.join(train_dir, category))):
            label=label_img(category)
            path=os.path.join(train_dir,category,img)
            img=cv2.imread(path,cv2.IMREAD_GRAYSCALE)
            img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
            train.append([np.array(img),np.array(label)])
    shuffle(train)
    return train
```

In [6]:

```
train_data = create_train_data() #creating training data
```

```
100%|████████████████████████████████████████| 263/263 [00:12<00:00, 21.35it/s]
100%|████████████████████████████████████████| 390/390 [00:08<00:00, 48.02it/s]
100%|████████████████████████████████████████| 287/287 [00:02<00:00, 97.12it/s]
100%|████████████████████████████████████████| 611/611 [00:05<00:00, 106.33it/s]
100%|████████████████████████████████████████| 221/221 [00:04<00:00, 44.26it/s]
100%|████████████████████████████████████████| 475/475 [00:05<00:00, 84.02it/s]
100%|████████████████████████████████████████| 654/654 [00:21<00:00, 30.64it/s]
100%|████████████████████████████████████████| 221/221 [00:06<00:00, 32.89it/s]
100%|████████████████████████████████████████| 516/516 [00:06<00:00, 83.34it/s]
100%|████████████████████████████████████████| 231/231 [00:02<00:00, 86.12it/s]
100%|████████████████████████████████████████| 496/496 [00:05<00:00, 88.68it/s]
100%|████████████████████████████████████████| 385/385 [00:12<00:00, 30.85it/s]
```

In [7]:

```
'''function that will create test data , will go thought file do this
----read the image in  grayscale mode ,resize it
---change it to numpy arrays and  append it to dataframe test but no category her

def create_test_data():
    test = []
    for img in tqdm(os.listdir(test_dir)):
        path = os.path.join(test_dir,img)
        img_num = img
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        test.append([np.array(img), img_num])

    shuffle(test)
    return test
```

In []:

```
In [8]: test_data = create_test_data() #creating test data
```

100%|██████████████████████| 794/794 [00:05<00:00, 134.20it/s]

```

In [9]: import tensorflow as tf #used for machine Learning applications such as neural ne
import tflearn #modular and transparent deep Learning library built on top of Ter
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 12, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy')

model = tflearn.DNN(convnet, tensorboard_dir='log')

if os.path.exists('{}.meta'.format(MODEL_NAME)):
    model.load(MODEL_NAME)
    print('model loaded!')

train = train_data
test = train_data

X = np.array([i[0] for i in train]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
test_y = [i[1] for i in test]

```

curses is not supported on this machine (please install/reinstall curses for an optimal experience)

WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from tensorflow.pytho

```

n.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-packages\tflearn\ini
tializations.py:119: UniformUnitScaling.__init__ (from tensorflow.python.ops.
init_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.initializers.variance_scaling instead with distribution=uniform to get
equivalent behavior.
WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-packages\tflearn\lay
ers\core.py:239: calling dropout (from tensorflow.python.ops.nn_ops) with kee
p_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - k
eep_prob`.
WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-packages\tflearn\obj
ectives.py:66: calling reduce_sum_v1 (from tensorflow.python.ops.math_ops) wi
th keep_dims is deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-packages\tensorflow
\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops)
is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

```

```

In [15]: model.fit({'input': X}, {'targets': Y}, n_epoch=3, validation_set=({'input': test
        snapshot_step=500, show_metric=True, run_id=MODEL_NAME)

```

```

Training Step: 2774 | total loss: 0.35367 | time: 108.915s
| Adam | epoch: 037 | loss: 0.35367 - acc: 0.8761 -- iter: 4736/4750
Training Step: 2775 | total loss: 0.35427 | time: 153.942s
| Adam | epoch: 037 | loss: 0.35427 - acc: 0.8791 | val_loss: 0.28483 - val_ac
c: 0.8973 -- iter: 4750/4750
--

```

```

In [16]: model.save(MODEL_NAME)

```

```

INFO:tensorflow:C:\Users\DELL\Desktop\nikhil project\plantclassification-0.001-2
conv-basic.model is not in all_model_checkpoint_paths. Manually adding it.

```

```
In [18]: #return Indexes of the maximal elements of a array
def label_return (model_out):
    if np.argmax(model_out) == 0: return 'Black-grass'
    elif np.argmax(model_out) == 1: return 'Charlock'
    elif np.argmax(model_out) == 2: return 'Cleavers'
    elif np.argmax(model_out) == 3: return 'Common Chickweed'
    elif np.argmax(model_out) == 4: return 'Common wheat'
    elif np.argmax(model_out) == 5: return 'Fat Hen'
    elif np.argmax(model_out) == 6: return 'Loose Silky-bent'
    elif np.argmax(model_out) == 7: return 'Maize'
    elif np.argmax(model_out) == 8: return 'Scentless Mayweed'
    elif np.argmax(model_out) == 9: return 'Shepherds Purse'
    elif np.argmax(model_out) == 10: return 'Small-flowered Cranesbill'
    elif np.argmax(model_out) == 11: return 'Sugar beet'
```

```
In [20]: import matplotlib.pyplot as plt

test_data = create_test_data()
fig=plt.figure(figsize = (18,10))
for num,data in enumerate(test_data[:12]):
    img_num = data[1]
    img_data = data[0]
    y = fig.add_subplot(3,4,num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)
    model_out = model.predict([data])[0]
    str_label=label_return (model_out)
    y.imshow(orig,cmap='gray',interpolation='nearest')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```


