UDACITY

‹ Return to "AI Programming with Python Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Create Your Own Image Classifier

REVIEW

CODE REVIEW 5

HISTORY

▸ train.py 2

▸ functions.py 2

▾ predict.py 1

```
__author__ = "Chris"

import argparse

import numpy as np
import torch.nn.functional as F
from torch import nn, optim
from torchvision import datasets, transforms, models
from PIL import Image
import json

from utility import load_data, process_image
from functions import load_checkpoint, predict, test_model

parser = argparse.ArgumentParser(description='Predict class of image with neura
# Argument for image path to be checked
parser.add_argument('--image_path',
                    action='store',
                    default='../aipnd-project/flowers/test/1/image_06743',
                    help='Path to image')
```

```
# Argument to store checkpoint
parser.add_argument('--save_dir',
                    action='store',
                    dest='save_directory',
                    default='checkpoint.pth',
                    help='Location to save checkpoint')
# Specify argument for pretrained neural network
parser.add_argument('--pretrain',
                    action='store',
                    dest='pretrained_model',
                    default='vgg11',
                    help = 'Pretrained model to implement; defaults to VGG-11;
                    can work with VGG and Densenet architectures')
# Specifiy argument for most likely classes of image
parser.add_argument('--top_k',
                    action='store',
                    dest='top_k',
                    type=int,
                    default=3,
                    help='Number of most likely classes to view; \
                         default 3; int type')
```

AWESOME

- Nice job parsing in the value ok K by the user

```
# Specify argument for image category
parser.add_argument('--cat_to_name',
                    action='store',
                    dest='cat_name_dir',
                    default='cat_to_name.json',
                    help='Path to image category')
# Specify argument for GPU mode
parser.add_argument('--gpu',
                    action='store_true',
                    default=False,
                    help='Turn GPU mode on; default False; \
                    bool type')
# Assign arguments
results = parser.parse_args()
save_dir = results.save_dir
image = results.image_path
top_k = results.top_k
cat_names = results.cat_name_dir
gpu = results.gpu
## Completion of argument assignment ##

with open(cat_names, 'r') as f:
    cat_to_name = json.load(f)

# Instantiate model
pret_model = results.pretrained_model
model = getattr(models, pret_model)(pretrained=True)

# Load model
loaded_model = load_checkpoint(model, save_dir, gpu)

# Preprocess image (w/ jpeg format)
```

```
processed_img = process_image(image)

if gpu == True:
    processed_img = processed_img.to('cuda')
else:
    pass

# Run prediction
probs, classes = predict(processed_img, loaded_model, top_k, gpu)
print(probs)
print(classes)

names = []
for i in classes:
    names += [cat_to_name[i]]

print(f"This flower is most likely to be a: '{names[0]}' with a probability of
```

▸ utilities.py

RETURN TO PATH