SmallSEOTools

PLAGIARISM SCAN REPORT

Words	383	Date	December 13,2020
Characters	4334	Exclude URL	

0%	100%	0	13
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

```
import matplotlib.pyplot as plt
import numpy as np
import torch
from torch import nn
from torchvision import datasets, transforms, models
from PIL import Image
import json
import argparse
parsers = argparse.ArgumentParser()
parsers.add_argument('data_dir', action="store")
parsers.add_argument('save_dir', action="store")
parsers.add_argument('--top_k', dest="top_k", default=5)
parsers.add_argument('--category_names', action="store", dest="category_names", default='cat_to_name.json')
parsers.add_argument('--gpu', action="store_const", dest="device", const="gpu", default='cpu')
arguments = parsers.parse_args()
data_dir = arguments.data_dir
save_dir = arguments.save_dir
top_k = arguments.top_k
category_names = arguments.category_names
device = arguments.device
with open(category_names, 'r') as f:
cat_to_name = json.load(f)
checkpoint = torch.load('checkpoint.pth')
checkpoint.keys()
model = models.vgg16(pretrained=True)
classifier = nn.Sequential( nn.Linear(25088, 450), nn.ReLU(), nn.Linear(450, 102)) #I got reference of syntax from
here:-> https://pytorch.org/docs/stable/generated/torch.nn.Sequential.html
## Ruberic:Loading checkpoints :-> There is a function that successfully loads a checkpoint and rebuilds the model
model.classifier = classifier
model.class_to_idx = checkpoint['class_to_idx']
model.load_state_dict(checkpoint['state_dict'])
path_to_image=data_dir
def process_image(path_to_image):
image = Image.open(path_to_image)
#Refrence is from https://pillow.readthedocs.io/en/latest/reference/Image.html
image=image.resize((image.width*256//image.height,256)) if image.width > image.height else
image.resize((256,image.height*256//image.width))
#in above expression,I have resize the images where the shortest side is 256 pixels, keeping the aspect ratio.
return ((((np.array(image.crop(((image.width-224)/2,(image.height-224)/2,(image.width-224)/2+224, (image.height-
```

```
224)/2+224))))/255) # image converted to np array from PIL
- np.array([0.485, 0.456, 0.406]))#substracting mean from resulting np array
/np.array([0.229, 0.224, 0.225])#Dividing by Standard Deviation
).transpose((2, 0, 1)) #transposing it to make third channel (color channel) to first and remaining both maintaining the sequence
def imshow(image, ax=None, title=None):
if ax is None:
fig, ax = plt.subplots()
image = image.numpy().transpose((1, 2, 0))
mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])
image = std * image + mean
image = np.clip(image, 0, 1)
ax.imshow(image)
return ax
def predict(image_path):
return (torch.exp(model.forward(
(torch.from_numpy( process_image(image_path)).type(torch.FloatTensor)).unsqueeze_(0) #Converting np array to float tensor using from_numpy
))).topk(5) #considering top 5 result refer from here https://pytorch.org/docs/master/torch.html#torch.topk
def display_img(image_path):
category = [] #Empty list
name_list=[] #Empty list
flower=[] #Empty list
imshow(process_image(image_path), plt.subplot(2,1,1)) #Displaying Image
probs, classes = predict(image_path) #predicting image with top 5 results
for i in classes[0]:
flower.append(i) #adding new element to flower
for x in flower:
for category_name, category_value in checkpoint['class_to_idx'].items(): #We are getting category key and value from class_to_idx dictonary
category.append(category_name) if category_value == x else continue #Storing Category Names if present in flower list
for i in category:
name_list.append(cat_to_name[i]) #if category value is present in cat_to_name then storing the label name to store it into name_list so that it would be use to print on bar chart
ax = plt.subplot(2,1,2)
ax.barh(name_list, probs[0]) #printing bar chart with possible naame_list and value
plt.show()
```

Sources	Similarity