

Supplement S5 File

January 29, 2019

1 Supplement S5 File

1.1 S5 File. Model testing.

```
In [ ]: #####
##### Script for testing a pytorch, convolutional neural net, using the pre-trained
##### resnet18 model #####
##### Authors: Hieu Le & Grant Humphries
##### Date: August 2018
##### This script was written for the Spacewhale project
##### and was based on the Pytorch transfer learning tutorial:
##### https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
#####
##### Usage examples (Linux)
#####
##### python testing_script.py
##### --data_dir /home/ghumphries/spacewhale/test --model MODEL1 --epoch 24
#####
#####
##### Setup information
##### To run this script, ensure that you have folders named exactly the same as
##### those in the training data folder. For example:
##### ./test/Water
##### ./test/Whale
##### IMPORTANT:
##### The images that you want to test should all live in the target folder.
##### For example, if you only want to test for water, then place all the
##### images in the ./test/Water folder. If you want to test for whales,
##### place all the images in the ./test/Whale folder
##### The data_dir argument should point to the directory ABOVE the training
##### folders. For example, if your directory is:
##### /home/user/spacewhale/testingdata/Water
##### then --data_dir /home/user/spacewhale/testingdata
#####
#####
### Library imports

from __future__ import print_function, division
```

```

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
from torchvision import datasets, models, transforms
from spacewhale_util import *
import os
import argparse

#####

### Create arguments for command line interface
parse = argparse.ArgumentParser()
parse.add_argument('--data_dir')
parse.add_argument('--model')
parse.add_argument('--epoch', type=int, default=24)
opt = parse.parse_args()

### Create the spacewhale class
s = spacewhale()

### Specify which epoch to load from the pre-trained model
epoch_to_use = 'epoch_'+str(opt.epoch)+'.pth'
trained_model = os.path.join('./trained_model', opt.model, epoch_to_use)

### Transform image to tensor for testing
test_transforms = s.data_transforms['test']

### Load the model into GPU:
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
torch.set_default_tensor_type('torch.cuda.FloatTensor')
model_ft = models.resnet18(pretrained=True)
num_fts = model_ft.fc.in_features
model_ft.fc = nn.Linear(num_fts, 2)
model_ft = model_ft.to(device)
model_ft.load_state_dict(torch.load(trained_model))

### Set the model into evaluation mode:
model_ft.eval()

## Data loader for the testing dataset
image_datasets = datasets.ImageFolder(opt.data_dir, s.data_transforms['test'])
dataloaders = torch.utils.data.DataLoader(image_datasets, batch_size=10,
                                           shuffle=False, num_workers=16)

### Run the model to predict all images in a directory
s.test_dir(device, model_ft, dataloaders)

```