Supplement S4 File

January 31, 2019

1 Supplement S4 File

1.1 S4 File. Model training.

```
In [ ]: ################################################################################
        #### Script for training a pytorch, convolutional neural net, using the pre-trained
        #### resnet18 model
        #### Authors:  Hieu Le, Grant Humphries, Alex Borowicz
        #### 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 training_script.py --name MODEL1 --data_dir /home/ghumphries/
        ####                           spacewhale/data --verbose True --epochs 19
        ####
        #################################################################################
        #### Setup information
        ####    To run this script, ensure that you have your training images inside of a
        ####     folder called 'train'.
        ####    Inside of the train folder, your images must be organized into folders based
        ####    on the label.  For example:
        ####    ./train/Water - this folder contains only water images in .png format
        ####    ./train/Whale - this folder contains only whale images in .png format
        ####    IMPORTANT:
        ####        The --data_dir argument must point to the folder ABOVE the 'train'
        ####         folder. For example:
        ####        .home/user/spacewhale/fulldata/train/... ->
        ####        data_dir usage:  --data_dir /home/user/spacewhale/fulldata
        ####
        ################################################################################
        ### 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
import matplotlib.pyplot as plt
import os
from spacewhale_util import *
import argparse
####################################################################################

### Create arguments for command line interface
parser = argparse.ArgumentParser()
parser.add_argument('--name',type=str)
parser.add_argument('--data_dir',type=str)
parser.add_argument('--verbose',type=bool,default=False)
parser.add_argument('--epochs',type=int,default=25)

opt = parser.parse_args()

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

### This creates a new directory called 'trained model' in the directory you are
### currently working from in Terminal
opt.checkpoint = ('./trained_model/'+opt.name)
s.sdmkdir(opt.checkpoint)

#Preparing the data
print('##############################################################################')
print('WELCOME TO SPACEWHALE!')
print('##############################################################################')
print('We will now train your model. Please be patient')
print('------------------------------------------------------------------------------')

### This part loads up any folders in the 'train' folder with the label being the
### name of the folder
image_datasets = {x: datasets.ImageFolder(os.path.join(opt.data_dir, x),
                                          s.data_transforms[x]) for x in ['train']}
weights = s.make_weights_for_balanced_classes(image_datasets['train'].imgs,
                                              len(image_datasets['train'].classes))
weights = torch.DoubleTensor(weights)
sampler = torch.utils.data.sampler.WeightedRandomSampler(weights, len(weights))
dataloaders = torch.utils.data.DataLoader(image_datasets['train'], batch_size=4,
                                          sampler = sampler, num_workers=4)
dataset_sizes = {x: len(image_datasets[x]) for x in ['train']}

print('Your dataset size is: %d'%(dataset_sizes['train']))
class_names = image_datasets['train'].classes
```

```
print('You have',str(len(class_names)),'classes in your dataset')

print('-----------------------------------------------------------------------------')
print('Labels for the dataset are:')
print(class_names[0] + ' = 0')
print(class_names[1] + ' = 1')
print('-----------------------------------------------------------------------------')
### This sets the device (if cuda is installed properly, it will send the
#### training data to the gpu)
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
dev = ("gpu" if torch.cuda.is_available() else "cpu")
print('Data loaded into', dev)
print('-----------------------------------------------------------------------------'

######################################################################################
### This part defines the model we're going to use
### First it downloads the pretrained resnet model (if we dont' have it) from modelZoo
### We count the number of features in the model and then replace the last layer with
### a linear layer so we can map our own classes. The model is sent to the gpu and we
### then opt to use CrossEntropy as the loss function. The optimizer is set as
### stochastic gradient descent with a learning rate of 0.001
### We then set the learning rate to decay every 7 epochs

model_ft = models.resnet18(pretrained=True)
num_ftrs = model_ft.fc.in_features
model_ft.fc = nn.Linear(num_ftrs, 2)
model_ft = model_ft.to(device)
criterion = nn.CrossEntropyLoss()
# Set Learning rate (lr) and step size below
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.0009, momentum=0.9)
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)

### If the verbose option is set, then print out the model
if opt.verbose:
    print(model_ft)

######################################################################################
### Run the train_model function from the spacewhale class

model_ft = s.train_model(opt, device, dataset_sizes, dataloaders, model_ft, criterion,
                         optimizer_ft, exp_lr_scheduler, num_epochs=opt.epochs)

######################################################################################
```