UDACITY

Back to AI Programming with Python Nanodegree

Create Your Own Image Classifier

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hi Student,

I literally enjoyed while reviewing you code. I really appreciate your efforts that you have kept in this project.

Additional Resources

- http://cs231n.github.io/transfer-learning/
- http://ruder.io/transfer-learning/

Congrats on finishing this project. Very good attempt at the first shot.

Student the effort that you have kept in this project never goes in vain. This project gives an opportunity to explore more because there is no straightforward solution for this. I can say the knowledge you gain by doing this project is really invaluable. Udacity is more of knowledge sharing and you can be master in a specific domain by completing the project, I want to see you out stand with the skills that you learn here.

Files Submitted

The submission includes all required files. (Model checkpoints not required.)

Part 1 - Development Notebook

All the necessary packages and modules are imported in the first cell of the notebook

Good job.

torchvision transforms are used to augment the training data with random scaling, rotations, mirroring, and/or cropping

I like the way you code is modularized and I am completely impressed with your code...

The training, validation, and testing data is appropriately cropped and normalized

Yes training data is even augmented.. Good job 

The data for each set (train, validation, test) is loaded with torchvision's ImageFolder

The data for each set is loaded with torchvision's DataLoader

A pretrained network such as VGG16 is loaded from torchvision.models and the parameters are frozen

Yes model has been loaded, and parameters have been freezed before fully connected layer. Good job 

A new feedforward network is defined for use as a classifier using the features as input

The parameters of the feedforward classifier are appropriately trained, while the parameters of the feature network are left static

During training, the validation loss and accuracy are displayed

Yes During training, the validation loss and accuracy are displayed

The network's accuracy is measured on the test data

Excellent job, wonderful effort.👍

There is a function that successfully loads a checkpoint and rebuilds the model

It has to be a function which will load parameters and rebuld the model.👍

The trained model is saved as a checkpoint along with associated hyperparameters and the class_to_idx dictionary

Good job by saving parameters.👍

The process_image function successfully converts a PIL image into an object that can be used as input to a trained model

The predict function successfully takes the path to an image and a checkpoint, then returns the top K most probably classes for that image

Great effort student.👍

A matplotlib figure is created displaying an image and its associated top 5 most probable classes with actual flower names

Part 2 - Command Line Application

train.py successfully trains a new network on a dataset of images and saves the model to a checkpoint

The training loss, validation loss, and validation accuracy are printed out as a network trains

The training script allows users to choose from at least two different architectures available from torchvision.models

The training script allows users to set hyperparameters for learning rate, number of hidden units, and training epochs

The training script allows users to choose training the model on a GPU

The predict.py script successfully reads in an image and a checkpoint then prints the most likely image class and it's associated probability

The predict.py script allows users to print out the top K classes along with associated probabilities

The predict.py script allows users to load a JSON file that maps the class values to other category names

The predict.py script allows users to use the GPU to calculate the predictions

DOWNLOAD PROJECT

RETURN TO PATH