UDACITY

‹ Back to AI Programming with Python Nanodegree

Create Your Own Image Classifier

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Brilliant Student,

The progress is wonderful and you met all the specifications this time around. Please keep the momentum going and you will definitely enjoy the remaining projects. Good luck for the rest of the Nanodegree.

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

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

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

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

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

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

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

The network's accuracy is measured on the test data

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

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

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

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