Create Your Own Image Classifier

Meets Specifications

Hey buddy. Good job building upon the comments from the previous reviewer. I can clearly see the effort that you have put in to complete this project, and also, your effort has paid off. I am happy to accept your submission as everything seems to work well! Great job!

Might I suggest 2 papers that I really think you should check out

- Flower Categorization using Deep Convolutional Neural Networks
- Recognition between a Large Number of Flower Species

And I also think you should read the following articles on the coding standards. You could really benefit from these.

- Top 15+ Best Practices for Writing Super Readable Code
- Good Programming Practices

Also, here is an optimal way to go about the entire problem. But just use it for reference and do not commit plagiarism.

I hope the following review comments are helpful for you, if they were, then I request you to please be generous with the ratings! :)

Files Submitted

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

Cleared by the previous reviewer.

Part 1 - Development Notebook

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

All the necessary packages and modules have been imported to the first cell of the notebook to clearly state the dependencies. Well done here!

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

✓ The network's accuracy is measured on the test data

Cleared by the previous reviewer.

✓ During training, the validation loss and accuracy are displayed

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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**

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

✓ **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 successfully reads in an image and a checkpoint then prints the most likely image class and it's associated probability. Good job!

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

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

Cleared by the previous reviewer.

DOWNLOAD PROJECT

RETURN TO PATH