Part 2 - Building the command line application

Now that you've built and trained a deep neural network on the flower data set, it's time to convert it into an application that others can use. Your application should be a pair of Python scripts that run from the command line. For testing, you should use the checkpoint you saved in the first part.

Specifications

The project submission must include at least two files `train.py` and `predict.py`. The first file, `train.py`, will train a new network on a dataset and save the model as a checkpoint. The second file, `predict.py`, uses a trained network to predict the class for an input image. Feel free to create as many other files as you need. Our suggestion is to create a file just for functions and classes relating to the model and another one for utility functions like loading data and preprocessing images. **Make sure to include all files necessary to run `train.py` and `predict.py` in your submission.**

- Train a new network on a data set with `train.py`
- Basic usage: `python train.py data_directory`
- Prints out training loss, validation loss, and validation accuracy as the network trains
- Options:
- Set directory to save checkpoints: `python train.py data_dir --save_dir save_directory`
- Choose architecture: `python train.py data_dir --arch "vgg13"`
- Set hyperparameters: `python train.py data_dir --learning_rate 0.01 --hidden_units 512 --epochs 20`
- Use GPU for training: `python train.py data_dir --gpu`
- Predict flower name from an image with `predict.py` along with the probability of that name. That is, you'll pass in a single image `/path/to/image` and return the flower name and class probability.
- Basic usage: `python predict.py /path/to/image checkpoint`
- Options:
- Return top *KK* most likely classes: `python predict.py input checkpoint --top_k 3`
- Use a mapping of categories to real names: `python predict.py input checkpoint --category_names cat_to_name.json`
- Use GPU for inference: `python predict.py input checkpoint --gpu`

The best way to get the command line input into the scripts is with the argparse module in the standard library. You can also find a nice tutorial for argparse here.