UDACITY

Back to AI Programming with Python Nanodegree

Image Classifier Application

REVIEW

CODE REVIEW 8

HISTORY

Meets Specifications

Awesome job completing the project!

Wish you the best in your future AI endeavors!

Here are some additional resources that provide a good overview of what we've learned 🚀:

- Comparison of Deep Learning Models for Image Classification
- Summary of the current state of Image Classification

Files Submitted

The submission includes all required files. (Model checkpoints not required.)
Well done submitting all of the required files!

Part 1 - Development Notebook

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

Well done organizing all of the import statements on the first cell of the notebook.

You could check out this thread to learn more about some of the advantages of doing so.

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

Nicely done augmenting the training data by randomly rotating, cropping and flipping!

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

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

Good job using the torchvision's ImageFolder, a generic data loader, to load the images!

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

Well done using a DataLoader to load the data for each set, which has the following benefits:

- Batching the data
- Shuffling the data
- Loading the data in parallel using multiprocessing workers.

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

Nice job loading the DenseNet121 model and freezing the parameters!

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

A new feedforward network has been defined to be used as a classifier using input features

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

Only the feedforward classifier is being appropriately trained while the feature network parameters are left static.

During training, the validation loss and accuracy are displayed

Well done clearly logging the validation loss and accuracy at each step!

The network's accuracy is measured on the test data

Awesome job achieving an accuracy of 92.06% on the test dataset!

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

Nicely done writing the `load_checkpoint` method to successfully load the checkpoint and rebuild the model

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

Great job saving the major hyperparameters in the checkpoint! This practice will make it easy to retrain your model in the future!

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

Awesome job finding the top K classes along with the associated probabilities!

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

8 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review

Student FAQ