U UDACITY

PROJECT SPECIFICATION

Create Your Own Image Classifier

Files Submitted

CRITERIA	MEETS SPECIFICATIONS
Submission Files	The submission includes all required files. (Model checkpoints not required.)

Part 1 - Development Notebook

CRITERIA	MEETS SPECIFICATIONS
Package Imports	All the necessary packages and modules are imported in the first cell of the notebook
Training data augmentation	torchvision transforms are used to augment the training data with random scaling, rotations, mirroring, and/or cropping
Data normalization	The training, validation, and testing data is appropriately cropped and normalized

CRITERIA	MEETS SPECIFICATIONS
Data batching	The data for each set is loaded with torchvision's DataLoader
Data loading	The data for each set (train, validation, test) is loaded with torchvision's ImageFolder
Pretrained Network	A pretrained network such as VGG16 is loaded from torchvision.models and the parameters are frozen
Feedforward Classifier	A new feedforward network is defined for use as a classifier using the features as input
Training the network	The parameters of the feedforward classifier are appropriately trained, while the parameters of the feature network are left static
Testing Accuracy	The network's accuracy is measured on the test data
Validation Loss and Accuracy	During training, the validation loss and accuracy are displayed
Loading checkpoints	There is a function that successfully loads a checkpoint and rebuilds the model

CRITERIA	MEETS SPECIFICATIONS
Saving the model	The trained model is saved as a checkpoint along with associated hyperparameters and the class_to_idx dictionary
Image Processing	The process_image function successfully converts a PIL image into an object that can be used as input to a trained model
Class Prediction	The predict function successfully takes the path to an image and a checkpoint, then returns the top K most probably classes for that image
Sanity Checking with matplotlib	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

CRITERIA	MEETS SPECIFICATIONS
Training a network	train.py successfully trains a new network on a dataset of images and saves the model to a checkpoint
Training validation log	The training loss, validation loss, and validation accuracy are printed out as a network trains

CRITERIA	MEETS SPECIFICATIONS
Model architecture	The training script allows users to choose from at least two different architectures available from torchvision.models
Model hyperparameters	The training script allows users to set hyperparameters for learning rate, number of hidden units, and training epochs
Training with GPU	The training script allows users to choose training the model on a GPU
Predicting classes	The predict.py script successfully reads in an image and a checkpoint then prints the most likely image class and it's associated probability
Top K classes	The predict.py script allows users to print out the top K classes along with associated probabilities
Displaying class names	The predict.py script allows users to load a JSON file that maps the class values to other category names
Predicting with GPU	The predict.py script allows users to use the GPU to calculate the predictions