UDACITY

‹ Return to "AI Programming with Python Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Create Your Own Image Classifier

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Dear student,
Congratulations on passing this project! 💯 👏 I believe you've learned a lot through this challenging project.

Awesome job! And here's an official tutorial of transfer learning from Pytorch https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html which might be helpful.

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

- Comparison of Deep Learning Models for Image Classification https://blog.paralleldots.com/data-science/must-read-path-breaking-papers-about-image-classification/
- Summary of the current state of Image Classification https://towardsdatascience.com/deep-learning-for-image-classification-why-its-challenging-where-we-ve-been-and-what-s-next-93b56948fcef
- Also, you can check how researchers solved this problem about 7 years ago: https://www.robots.ox.ac.uk/~vgg/research/flowers_demo/docs/Chai11.pdf

Wish you the best in your future AI endeavors!

Stay udacious and keep going!

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

Well done!
`torchvision transforms` are implemented for data augmentation, which will make the training process more robust and address the overfitting.

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

Nice!
All three datasets have been appropriately preprocessed.

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

Nice!
`DataLoader` has been implemented to prepare train_loaders, valid_loaders, and test_loaders for further loading the data.

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

Good!
The pre-trained network `VGG` has been loaded and the parameters have been frozen.

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

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

Good job!
`ImageFolder` has been used for preparing the datasets.

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

Nice! The code for training and validation is well organized.

The network's accuracy is measured on the test data

During training, the validation loss and accuracy are displayed

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

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

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

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

Nice! You code 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 choose training the model on a GPU

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

Good job! You've designed an interface for users.

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

Nice! 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 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

Rate this review