Use a Pre-trained Image Classifier to Identify Dog Breeds

REVIEW

CODE REVIEW 7

HISTORY

Meets Specifications

Hello Karim Sakr,
👏👏👏 Congratulations on completing this project successfully.
I must admit that the structure of this project implementation is good and worth each scroll down your code.
To say that I enjoyed going through your work would be an understatement.

Specifically, **your code is clean, easy to follow and very well commented.** This is very impressive.

You obviously put a lot of work into this and I am glad that it paid off.
The results obtained from your models clearly proves how excellent your implementation was.
Keep up the good work! ✊
Be Udacioused and Keep the it going like this!

Congratulations and wish you all the best with the rest of the nanodegree.

Finally, I would like to share with you some great articles that highlights key concepts for learning Python programming:

- Learning Python: From Zero to Hero
- 30 Python Best Practices, Tips, And Tricks
- Classify Images Using Convolutional Neural Networks & Python

Stay safe 😄

Timing Code

✓

Student calls the time functions before the start of main code and after the main logic has been finished.

✅ Nice to see you use python's time function to time your code.

Great job measuring the code execution time. This is the first step in the process of code optimization. This is very useful to check the performance of your project, especially as it scales in size and complexity.

ADDITIONAL READINGS ON TIME FUNCTIONS

1. Time Functions in Python
2. Time module and sleep function
3. Time access and conversions

Keep it up!

Command Line arguments

✓

adds command line argument for '--dir'
uses default ='pet_images/'

Great use of python's `argparse` module to create your command line arguments.
✅ The output of the `python check_images.py --dir pet_images/` is excellent. I was able to test use several directories without issues.

- The purpose of command line arguments is to provide a way for your programs to be more flexible by allowing external inputs (command line arguments) to be input into a program. Adding the --dir CLI allows working directory flexibility and not always to a specific directory.

EXTRA READINGS ON CLI

1. The Easy Guide to Python Command Line Arguments
2. Parser for command-line options, arguments and sub-commands

✓

adds command line argument for '--arch'
default='vgg'

- ✅ Great to see the your implementation `---arch` works perfect with `default='vgg'`
  Kudos 👍

✓

adds command line argument for '--dogfile'
default='dognames.txt'

Excellent approach demonstrated here in implementing the command-line arguments.

- ✅ `python check_images.py --help` gives a good results with the information of with arguments are being passed.
- ✅ Great idea specifying the `default` values and `types`, when no command-line argument is entered when running `python check_images.py`, the default values are used.

Pet Image Labels

✓

Makes sure files starting with '.' are ignored.
Checks for '.' using a conditional statement.

Excellent approach to check and skip hidden files in directories.

- ✅ Good work ignoring filenames starting with `'.'`
- Do you know, it's possible to pass in a `Tuple` to `str.startswith()` ?. Kindly visit programiz short tutorial on `startswith()` for further insights.

✓

Dictionary key and label are in the correct format and retrieves 40 key-value pairs.
e.g:- {'Poodle_07956.jpg': ['poodle'], 'fox_squirrel_01.jpg': ['fox squirrel'] ... }

Brilliant job building the pet image label dictionary.

- ✅ You have correctly returned all key-value pairs and label in the correct format.Great showmanship of string manipulation.
- For more information on dictionaries, check Python | Ways to create a dictionary of Lists and Dictionaries in Python

✓

'in_arg.dir' is passed as an argument inside check_images.py while calling the get_pet_labels function.

Excellent! By doing this, the argument value of `--dir` is retrieved from the user to the `get_pet_labels` function.

- ✅ Nice implementation of the `get_pet_labels()` function.
- By doing this, you give yourself the ability to use the command line arguments you created earlier

Classifying Images

✓

Appends images_dir to each value before making the function call.

classifier(images_dir+users_key, model)

Well implemented!

- ✅ You correctly concatenated `images_dir` and `key` which represent the full path to each pet image file.
- The image directory is a choice of the user and changes often. It is thus a great idea to separate the filenames from the image directory and later concatenate the two to form a relative path to the files

✓

Convert the output to lower case and strip any whitespaces

Excellent work!

- ✅ Good job in applying `lower()` and `strip()` to the classifier output.
- This will help ensure uniformity throughout the code and importantly to accurately compare the classifier output with the pet label efficiently.

✓

Appends 1 to correct label, and 0 to falsely classified values

Using the `extend()` function is the most concise way again of doing this, as you know it is used to concatenate two lists together, so having the items packed in a list, allows them to be added in one step to the end of the lists of values of `results_dic` well done!

Classifying Labels as Dogs

✓

Check the displayed output and see if all matches are appropriately displayed.

All matches appropriately displayed in the display output are correctly classified as either "dogs" or "not dogs". Well done!. Good job!

✓

Check the displayed output and see if all non matches are appropriately displayed

All non-matches appropriately displayed in the display output are correctly classified as either "dogs" or "not dogs". Well done!. Good job!

Results

All three models score as expected.

Well done with all models' scores.

RESNET

```
Results for Resnet CNN model architecture.

Number of images: 40
Number of dog images: 30
Number of not dog images: 10

pct_match: 82.5
pct_correct_dogs: 100.0
pct_correct_breed: 90.0
pct_correct_notdogs: 90.0

Incorrectly Classified Dogs:
Real: cat                 , Classifier: norwegian elkhound, elkhound

Incorrectly Classified Breeds:
Real: great pyrenees      , Classifier: kuvasz
Real: beagle              , Classifier: walker hound, walker foxhound
Real: golden retriever    , Classifier: leonberg

** Total Elapsed Runtime: 0:0:5
```

VGG

```
Results for Vgg CNN model architecture.

Number of images: 40
Number of dog images: 30
Number of not dog images: 10

pct_match: 87.5
pct_correct_dogs: 100.0
pct_correct_breed: 93.33333333333333
pct_correct_notdogs: 100.0

Incorrectly Classified Breeds:
Real: great pyrenees      , Classifier: kuvasz
Real: beagle              , Classifier: walker hound, walker foxhound

** Total Elapsed Runtime: 0:0:33
```

ALEXNET

```
Results for Alexnet CNN model architecture.

Number of images: 40
Number of dog images: 30
Number of not dog images: 10

pct_match: 75.0
pct_correct_dogs: 100.0
pct_correct_breed: 80.0
pct_correct_notdogs: 100.0

Incorrectly Classified Breeds:
Real: beagle               , Classifier: english foxhound
Real: great pyrenees       , Classifier: kuvasz
Real: beagle               , Classifier: walker hound, walker foxhound
Real: boston terrier       , Classifier: basenji
Real: golden retriever     , Classifier: tibetan mastiff
Real: golden retriever     , Classifier: afghan hound, afghan

** Total Elapsed Runtime: 0:0:3
```

DOWNLOAD PROJECT

7 CODE REVIEW COMMENTS ›

RETURN TO PATH