Use a Pre-trained Image Classifier to Identify Dog Breeds

REVIEW | CODE REVIEW 4 | HISTORY

Meets Specifications

Congrats Isaak!!

Well done on completing all required tasks to pass and complete this project. Learning something new takes time and patience and you have proven to be determined. All the best as you progress through your coursework. More exciting challenges lie ahead but I believe you possess what it takes to get through. In the next project, you will focus on building your own classifier. Be ready! You will cover deep staff in AI.

Awesome using the extend list function to add items to the results dictionary simultaneously without the need to call *the* `append()` *function multiple times.*

Extra resources to keep learning

- Understanding Data Structures & Algorithms will further help you write efficient codes. *Lesson #7* will introduce you to *Big O Notation* & *Time Complexity.*
- String manipulation in python.
- Learn about `Argparser` positional and optional arguments here.
- Understanding time complexity with Python examples.

All the best through your coursework.
Stay safe.

Timing Code

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

Great job timing your code with the time function.

- Timing your program or a portion of your program's code, allows one to compare the *time costs* associated with using different algorithms to solve a problem. Additionally, timing your code allows one to know the time costs associated with running a program using given computing resources. *Time complexity* is the amount of **computational resources** *(time and memory)* it takes to run an algorithm. Take a look at **Time complexity** for further reading. In there, you will discover some classes of commonly encountered time complexities like *linear time* and *constant time.*
- This will be very useful to check your project's performance, especially as it scales in size and complexity. You can also learn more about the **Time module and sleep function** in your spare time.

Command Line arguments

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

Adding the `--dir` CLI allows working directory flexibility and not always to a specific directory.

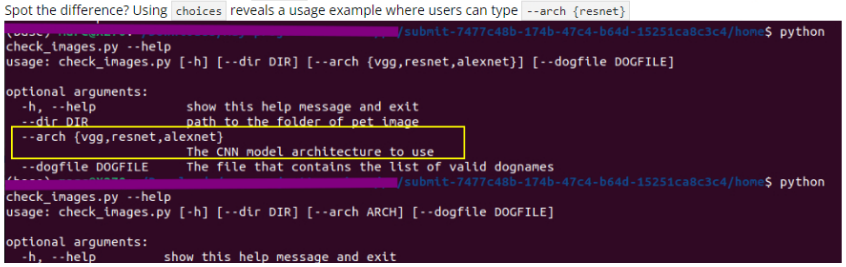
- `Argparser` implemented nicely, take a look at the differences between positional and optional arguments within this documentation for further insights.

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

Excellent, *VGG* is used as the default value of the `--arch` command-line argument. You can specifically inform users of the CNN model architecture your applications support by using the `choices` keyword argument of *add_argument* method to list the supported architectures. *As an ML engineer, you should always keep in mind your project will be used by others and it's, therefore, ideal to make interaction with your application easy and intuitive.*
The `choices` *keyword can be used like this:*

```
…
parser.add_argument("--arch"... choices=['vgg', 'alexnet', 'resnet'])
…
```

Below Figure shows when one uses the `choices` keyword to list supported models **(yellow box)** vs when one does not **(cyan box)**. Spot the difference? Using `choices` reveals a usage example where users can type `--arch {resnet}`



```
 --dir DIR          path to the folder of pet image
 --arch ARCH        The CNN model architecture to use
 --dogfile DOGFILE  The file that contains the list of valid dognames
```

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

Good! Command-line arguments are neatly defined and work just fine. All three arguments have meaningful `help` messages and their `type`s are also well defined. Kindly visit this documentation to learn more useful methods, this will come in handy in your final project.

Pet Image Labels

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

Hidden files (files starting with a '.') are filtered out in your program. Well done!

- These files can be useful for system configuration, considering them as conventional input files could cause potential *flaws* in your program as it scales.
- Another approach to check and skip '.' file(s) is by using Python's `str.startswith()` method and `continue` statement respectively. 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.

- All key-value pairs returned in the correct formats. This proves you have the skills to manipulate data to produce a given format. Well done!
- Further read on string manipulation in python to get familiarized with other useful built-in methods. Kindly ignore it if you have a solid grasp of the subject.

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

Classifying Images

✓ Appends images_dir to each value before making the function call.

classifier(images_dir+users_key, model)

Fantastic understanding, since the files are being accessed from the located directory, you need to concatenate the `images_dir` or *image directory* and `key` or *filename* strings to detect the correct relative path to the files.

✓ Convert the output to lower case and strip any whitespaces

Good work done! For consistency and uniformity in comparing labels: stripping off leading & trailing whitespace characters from the `classifier` output and converting them to lowercase will effectively help to compare the `classifier` output with the `pet_label`.

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

Classifying Labels as Dogs

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

Matches between classifier and pet Image labels correctly classify each label as "dogs" or "not dogs". Well done!

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

All Non-matches between *classifier_labels* and *pet_labels* are correctly classified as "dogs" or "not dogs".

Results

✓ All three models score as expected.

All results stats calculated as well as dog/not dog and dog breed classifications are correct. Well done!

```
*** Results Summary for CNN Model Architecture RESNET ***
N Images            :  40
N Dog Images        :  30
N Not-Dog Images    :  10

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

INCORRECT Dog/NOT Dog Assignments:
Real:                     cat   Classifier:   norwegian elkhound, elkhound

INCORRECT Dog Breed Assignment:
Real:           golden retriever   Classifier:                     leonberg
Real:             great pyrenees   Classifier:                       kuvasz
Real:                     beagle   Classifier:  walker hound, walker foxhound

** Total Elapsed Runtime: 0:0:2
```

```
*** Results Summary for CNN Model Architecture ALEXNET ***
N Images            :  40
N Dog Images        :  30
N Not-Dog Images    :  10
```

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

INCORRECT Dog Breed Assignment:
Real:                   beagle   Classifier:              english foxhound
Real:           golden retriever   Classifier:               tibetan mastiff
Real:             great pyrenees   Classifier:                       kuvasz
Real:                   beagle   Classifier:  walker hound, walker foxhound
Real:           golden retriever   Classifier:          afghan hound, afghan
Real:             boston terrier   Classifier:                       basenji

** Total Elapsed Runtime: 0:0:1
```

```
*** Results Summary for CNN Model Architecture VGG ***
N Images            :  40
N Dog Images        :  30
N Not-Dog Images    :  10

pct_match           :  87.5
pct_correct_dogs    : 100.0
pct_correct_breed   :  93.3
pct_correct_notdogs : 100.0

INCORRECT Dog Breed Assignment:
Real:             great pyrenees   Classifier:                       kuvasz
Real:                   beagle   Classifier:  walker hound, walker foxhound

** Total Elapsed Runtime: 0:0:18
```

DOWNLOAD PROJECT

4 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review

START