**Image Classification for a City Dog Show**

### **Use a Pre-trained Image Classifier to Identify Dog Breeds**

# Project Description:

**Project Goal**

* Improving your *programming* skills using Python.

In this project, you will use a *created* image classifier to identify dog breeds. We ask you to focus on *Python* and **not** on the actual classifier.

**Description:**

Your city is hosting a citywide dog show and you have volunteered to help the organizing committee with contestant registration. Every participant that registers must submit an image of their dog along with biographical information about their dog. The registration system tags the images based upon the biographical information.

Some people are planning on registering pets that **aren’t actual dogs**.

You need to use an already developed Python classifier to make sure the participants are dogs.

**\*Note, you DO NOT need to create the classifier. It will be provided to you. You will need to apply the Python tools you just learned to USE the classifier**.

**Your Tasks:**

* Using your Python skills, you will determine which image classification algorithm works the "**best**" on classifying images as "dogs" or "*not* dogs".
* Determine **how well** the "best" classification algorithm works on correctly identifying a dog's breed. If you are confused by the term *image classifier* look at it simply as a tool that has an input and an output. The Input is an image. The output determines what the image depicts. (for example, a dog). Be mindful of the fact that image classifiers do not always categorize the images correctly.
* **Time** how long each algorithm takes to solve the classification problem. With computational tasks, there is often a trade-off between accuracy and runtime. The more accurate an algorithm, the higher the likelihood that it will take more time to run and use more computational resources to run.

For further clarifications, please check our [FAQs](https://github.com/udacity/AIPND-revision/blob/master/notes/project_intro-to-python.md) here.

**Important Notes:**

For this image classification task, you will be using an image classification application using a deep learning model called a convolutional neural network (often abbreviated as CNN). CNNs work particularly well for detecting features in images like colors, textures, and edges; then using these features to identify objects in the images. You'll use a CNN that has already *learned* the features from a giant dataset of 1.2 million images called [ImageNet](http://www.image-net.org/). There are different types of CNNs that have different structures (architectures) that work better or worse depending on your criteria. With this project, you'll explore the three different architectures (**AlexNet**, **VGG**, and **ResNet**) and determine which is best for your application.

We have provided you with a ***classifier function*** in classifier.py that will allow you to use these CNNs to classify your images. The test\_classifier.py file contains an example program that demonstrates how to use the ***classifier function***. For this project, you will be focusing on using your Python skills to complete these tasks using the ***classifier function.***

Remember that certain breeds of dogs look very similar. The more images of two similar-looking dog breeds that the algorithm has *learned* from, the more likely the algorithm will be able to distinguish between those two breeds. We have found the following breeds to look very similar: [Great Pyrenees](https://www.google.com/search?q=Great+Pyrenees&source=lnms&tbm=isch&sa=X&ved=0ahUKEwje252-kpfZAhVF3FMKHeXwB3IQ_AUICigB&biw=1112&bih=1069) and [Kuvasz](https://www.google.com/search?tbm=isch&q=Kuvasz&spell=1&sa=X&ved=0ahUKEwi9_9fTkpfZAhWB7FMKHXlKDWoQBQg6KAA&biw=1112&bih=1069&dpr=1" \t "_blank), [German Shepherd](https://www.google.com/search?biw=1112&bih=1069&tbm=isch&sa=1&ei=d7F8WpaaMc_VzgLW8LvABw&q=German+Shepherd&oq=German+Shepherd&gs_l=psy-ab.3..0i67k1j0l2j0i67k1j0l6.31751.41069.0.41515.29.18.4.7.9.0.131.1164.14j2.17.0....0...1c.1.64.psy-ab..2.26.1140.0..0i10k1j0i13k1.112.xUB8_AoVF9w) and [Malinois](https://www.google.com/search?biw=1112&bih=1069&tbm=isch&sa=1&ei=orF8WtHWDcOdzwLnyLXgBw&q=Malinois&oq=Malinois&gs_l=psy-ab.3..0l3j0i67k1l3j0l2j0i67k1j0.31864.42125.0.42493.23.20.0.1.1.0.132.1460.14j4.19.0....0...1c.1.64.psy-ab..8.14.926.0...75.U5aOu6JZ9Vk" \t "_blank), [Beagle](https://www.google.com/search?biw=1112&bih=1069&tbm=isch&sa=1&ei=zbF8WqTiHZDxzgKlm5SYBw&q=Beagle&oq=Beagle&gs_l=psy-ab.3..0i67k1j0l2j0i67k1l2j0l5.29396.33482.0.34041.12.8.3.1.1.0.126.585.6j2.8.0....0...1c.1.64.psy-ab..0.12.609...0i10k1.0.Dr92CW2Kqqo) and [Walker Hound](https://www.google.com/search?biw=1112&bih=1069&tbm=isch&sa=1&ei=8LF8WteAGND0zgKvlL-IBw&q=Walker+hound&oq=Walker+hound&gs_l=psy-ab.3..0l10.20697.23454.0.23773.12.10.0.2.2.0.81.601.10.10.0....0...1c.1.64.psy-ab..0.12.610...0i67k1.0.GI0QxI1sadY), amongst others.

# Project Instructions:

## ****Principal Objectives****

1. Correctly identify which pet images are of dogs (even if the breed is misclassified) and which pet images aren't of dogs.
2. Correctly classify the breed of dog, for the images that are of dogs.
3. Determine which CNN model architecture (ResNet, AlexNet, or VGG), "**best**" achieve objectives 1 and 2.
4. Consider the time resources required to best achieve objectives 1 and 2, and determine if an alternative solution would have given a "good enough" result, given the amount of time each of the algorithms takes to run.

## ****TODO:****

### Edit program *check\_images.py*

The ***check\_images.py*** is the program file that you will be editing to achieve the four objectives above. This file contains a main() function that outlines how to complete this program through using functions that have **not** yet been defined. You will be creating these undefined functions in **check\_images.py** to achieve the objectives above.

All of the TODOs are listed in check\_images.py. You will find further elaborations and explanations for each, in the following concepts of this project.

## \*If you feel that you need more guidance, please refer to the files ending with\_hints.py. In the workspace, you will find a hint file for each of the tasks.

### **Important notes:**

* The **Project Workspace** is set up with the programs and files (like pet\_images folder) you will need to complete the project.
* The Python comments that begin with ***# TODO:*** in the **check\_images.py** program indicates where you will need to change the code of the program. The comments in **check\_images.py** will help you make the changes needed.
* Function docstrings contain input **parameters** and **return values**, which were left to provide guidance. You are welcome to program these functions differently.
* In **6. Timing Code** to **19. Printing Results** we will provide additional guidance for programming the undefined functions and completing the **check\_images.py** program. This information has been provided to help you through the process. The information provides:
* Which **Lessons** to review regarding programming the undefined functions.
* Details about the assignment's files (e.g. image files in pet\_images folder, dognames.txt).
* Details regarding using the classifier function in **classifier.py**.
* Links to relevant python documentation.
* Relevant example code.
* You can use the functions within the program [print\_functions\_for\_lab\_checks.py](https://github.com/udacity/AIPND-revision/blob/master/intropyproject-classify-pet-images/print_functions_for_lab_checks.py) to check your code for sections **8. Command Line Arguments** through **17. Calculating Results**. You will find this program within the Project Workspace.

## Program Outline

* Time your program

Use Time Module to compute program runtime

* Get program Inputs from the user

Use command line arguments to get user inputs

* Create Pet Images Labels

Use the pet images filenames to create labels

Store the pet image labels in a data structure (e.g. dictionary)

* Create Classifier Labels and Compare Labels

Use the Classifier function to classify the images and create the classifier labels

Compare Classifier Labels to Pet Image Labels

Store Pet Labels, Classifier Labels, and their comparison in a complex data structure (e.g. dictionary of lists)

* Classifying Labels as "Dogs" or "Not Dogs"

Classify all Labels as "Dogs" or "Not Dogs" using dognames.txt file

Store new classifications in the complex data structure (e.g. dictionary of lists)

* Calculate the Results

Use Labels and their classifications to determine how well the algorithm worked on classifying images

* Print the Results

You will need to repeat these tasks for each of the three image classification algorithms that are provided to you.