**Arden Ma**

### 7/4/2019

Engineers for Exploration

**Using annotate\_cpu.py and annotate\_gpu.py**

# INTRODUCTION

## PREREQUISITES

TensorFlow 1.14 and Python 2.7 (CPU)

TensorFlow-GPU 1.14 and Python 3.5 (GPU)

Frozen model graph generated from retrain.py

# NOTES

One of the files that is necessary in this tutorial is an already trained model from using retrain.py. This is a file that is given after the training process is complete and this graph is necessary in order to do the classification, and if you haven’t done this step, please refer to the *TensorFlow - Using retrain.py* documentation.

With this model, download the annotate\_cpu.py script if using CPU, otherwise if a GPU is available, download annotate\_gpu.py from the mangrove monitoring Github under Tools.

## 

## annotate.py

The first step in the process of getting the real world accuracy of your model is to classify your images first. This is done using annotate.py (CPU or GPU version). The functionality and arguments of both versions are the same, except for the additional batch size argument for the GPU version, so this documentation should suffice as explanation for how to use both scripts. These programs are a modified version of TensorFlow’s label\_image.py, and can classify entire directories of images and print the results. These file will take an input of the directory containing your images, your graph generated from the training, a labels file, the name and location of your result file, and finally a batch size if running the GPU version. Make sure to be in an environment (venv) containing the correct TensorFlow and Python versions.

Preferably organize your testing data as such, with images inside their own folders representing the names of the classes, with the naming convention of classname\_imagename.jpg of the images inside.

│ images  
└───classname\_1  
│ │ classname1\_image1.jpg  
│ │ classname1\_image2.jpg  
└───classname\_2  
 │ classname2\_image1.jpg  
 │ classname2\_image2.jpg

The list of arguments and their explanations are as follows:

* -h
  + This argument lists all of the arguments that are covered in this documentation file and smaller
* **--images (directory)**
  + This is a directory of images that you want to test the accuracy of. Use the organization convention above to help with organizing your classifications.
* **--graph (file)**
  + This is the output graph generated from retrain.py. The default location for the output graph is is /tmp/output\_graph.pb, however you can change this with the --output\_graph argument in retrain.py.
* **--labels (file)**
  + This is a file that lists the label names used to classify these images. The location of this file is in /tmp/output\_labels.txt.
* **--output\_file (file)**
  + This is the filename/filepath where you would like to store the results of the script (output class proababilities for each image).
* **--batch\_size (int) \*\*\* GPU version only \*\*\***
  + This is the image batch size that the GPU will process. Do not include this argument for the CPU version. On a single 1080Ti, 512 performed well as a batch size (1024 resulted in OOM).
* --input\_height (int)
  + This is the input height of the images into the classifier. The default and maximum is 299 pixels
* --input\_width (int)
  + This is the input width of the images into the classifier. The default and maximum is also 299 pixels.
* --input\_mean (float)
  + This argument is the input mean of the bands in the input images. This is used to normalize the input images to make classification easier.
* --input\_std (float)
  + This is the standard deviation of the bands in the input images. This is also used to normalize the input images.
* **--input\_layer (str)**
  + This is the name of the input layer of your model. By default this is simply ‘Placeholder’ if you do not change it, which you typically cannot.
* **--output\_layer (str)**
  + This is the name of the final layer of your model. By default this is ‘final\_result’, but this can be changed by the --final\_tensor\_name argument in retrain.py

An example of a correct use of annotation.py follows:

sudo python annotate.py --images ‘~/mangrove/images/red/’ --graph ‘/tmp/output\_graph.pb’ --labels ‘/tmp/output\_labels.txt’ –-output\_file ‘/mangrove/results/results\_file.txt’ --batch\_size 512 –-input\_layer=Placeholder --output\_layer=final\_result

After annotate.py is ran, every image should be classified with these classifications with these results put in the current directory of the annotate.py program. This program outputs the file name of the image that it classified and then the confidence of its classifications in the next lines. The output should look like this:

result\_file.txt  
│ classname1\_image1.jpg  
│ class1 confidence\_level  
│ class2 lower\_confidence\_level  
└ ...

# REFERENCES

https://github.com/UCSD-E4E/mangrove/blob/master/Documentation/E4E%20Documentation\_%20Using%20annotate.py%20and%20accuracy.py\_.docx

https://www.tensorflow.org/hub/tutorials/image\_retraining

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/label\_image/label\_image.py#L20

https://github.com/tensorflow/hub/blob/master/examples/image\_retraining/retrain.py