**Dillon Hicks**

### 8/20/2018

Engineers for Exploration

**TensorFlow - Using retrain.py**

# INTRODUCTION

## OVERVIEW

Using TensorFlow in combination with it’s hub module can allow you to quickly and easily create CNN models using the vast amount of architectures that exist on the TensorFlow Hub. This document will teach you how to use the retrain.py file and each of it’s arguments.

## PREREQUISITES

TensorFlow 1.10

TensorFlow Hub

retrain.py

# NOTES

The advantage of transfer learning is that you don’t need to retrain the entire model. Oftentimes this takes millions of images, and a very large amount of processing powers, but not all problems require this type of intensity. With this, only the last layer of the network will be trained, using the training on previous datasets to be used in the other layers. This can be effective for most tasks with a limited number of classes, as these pretrained layers are useful for most classification tasks. The retrain.py program takes arguments in the form of --argument\_name parameter. Each argument is listed The arguments that you should use during every training or should use to make your classification easier are bolded below for easier access. In addition, images are given to see the effect that these options have on the training images. The original image is in red, with the modified image in green

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
* **--image\_dir (directory)**
  + This argument takes in a directory of the images you want to be processed. In this folder you should have each image class that you want to be represented in your model in each corresponding folder. For example, in the case of mangroves, with the three main classifications of red, white, and black mangroves:
* **--output\_graph (file)** 
  + This takes in an input of a directory of where you want to save the graph of your model
  + default='/tmp/output\_graph.pb'
* --intermediate\_store\_frequency (int)
  + This argument takes in an integer value and denotes how many seconds tensorflow should wait before storing a copy of the current output graph. This is useful when using checkpoints when you need to stop and resume your training.
  + default=0
* --intermediate\_output\_graphs\_dir (directory)
  + This is the directory in where the intermediate output graphs are saved for you to access them
  + default='/tmp/intermediate\_graph/'
* **--output\_labels (directory)**
  + This is where the output labels are outputted, it should output a text file which should look like:  
    output\_labels.txt  
    │ label1  
    │ label2  
    │ label3  
    └ ...
  + default='/tmp/output\_labels.txt'
* --summaries\_dir (directory)
  + This directory is where the summaries for the tensorboard are saved. Useful for seeing in retrospect the results and statistics of your training.
  + default='/tmp/retrain\_logs'
* **--how\_many\_training\_steps (int)**
  + Integer of how many training steps (epochs) you want to use in your training. Avoid using too many steps to avoid overfitting or too little to avoid underfitting. Also adjust this to avoid long training times for smaller datasets.
  + default=4000
* --learning\_rate (float)
  + This is a number between 0 and 1 that can define how quickly your model diverges it’s weights from their previous values. A too high learning rate can diverge quickly in it’s loss graph and can in fact can increase in loss. A too low learning rate can take too long to train and won’t converge closer to 0 loss than if you selected a higher learning rate. Choose this value carefully.
  + default=0.01
* --testing\_percentage (int)
  + This is the percentage (0-100) of images that is used to give an unbiased estimation of the accuracy of the model. This is on images that the model is not yet trained with, and thus can give a closer real world approximation to how the model will react to unknown images.
  + default=10
* --validation\_percentage (int)
  + This is the percentage of images that the model sees to tune it’s hyperparameters to get a more accurate classification. This is what the loss is calculated on, but these images are not used to train, only to see how the model fits the training dataset and how well the training is performing.
  + default=10
* --eval\_step\_interval (int)
  + This is the amount of steps (epochs) that the
  + default=10
* --train\_batch\_size (int)
  + This is the amount of images that are passed through the network every training step. As you increase batch size this decreases the amount of backpropagation through the network, decreasing the accuracy. Keep this value low, but a batch size too low can greatly increase the time it takes your model to train.
  + default=100
* --test\_batch\_size (int)
  + This is the amount of images that are tested in each epoch to test the current performance of the model. Use -1 to use the entire dataset.
  + default=100
* --validation\_batch\_size (int)
  + How many images to use in an evaluation batch. This validation set is used much more often than the test set, and is an early indicator of how accurate the model is during training. A value of -1 causes the entire validation set to be used, which leads to more stable results across training iterations, but may be slower on large training sets. (from retrain.py -h)
  + default=100
* --print\_misclassified\_images
  + This is whether to print the misclassified images to the console.
  + default=False
* --bottleneck\_dir (directory)
  + This is where you can store the bottlenecks that are generated after the
  + default='/tmp/bottleneck'
* --final\_tensor\_name (str)
  + This is the final name of the final layer in the retrained model
  + default=’final\_result’
* --flip\_left\_right
  + Flips the image left or right from its horizontal position

 

* + default=False
* --random\_crop (int)
  + Crops the image in a random position inside the image with the crop being a percentage size of the total image



* + default=0
* --random\_scale (int)
  + Crops the image to the center but crops to a random amount from the percentage to the full size of the image



* + default=0
* --random\_brightness (int)
  + Range to change the the brightness of the image randomly



* + default=0
* --tfhub\_module (str)
  + If you want to change the network architecture from the default of Inceptionv3 to another, enter the module name from which the architecture will be downloaded from the TensorFlow Hub. To find more modules visit: https://www.tensorflow.org/hub/modules/
* --saved\_model\_dir (directory)
  + This is where you will save your final model that has been trained from the images

# REFERENCES

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

https://www.tensorflow.org/hub/modules/

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