

IEEE QP Wiki

Engineer. Design. Innovate. Mentor. Showcase

TensorFlow



TensorFlow

This tutorial assumes that you already have a basic understanding of the concepts of machine learning. If you don't, a good start would be the machine learning crash course given by Google!

<https://developers.google.com/machine-learning/crash-course/>

In addition, this tutorial also assumes that you have a simple understanding of UNIX commands and how to run python executables.

In addition, you need to ensure that all of these are installed:

Python 3.7.x (along with pip):

<https://www.python.org/downloads/>

TensorFlow 1.10

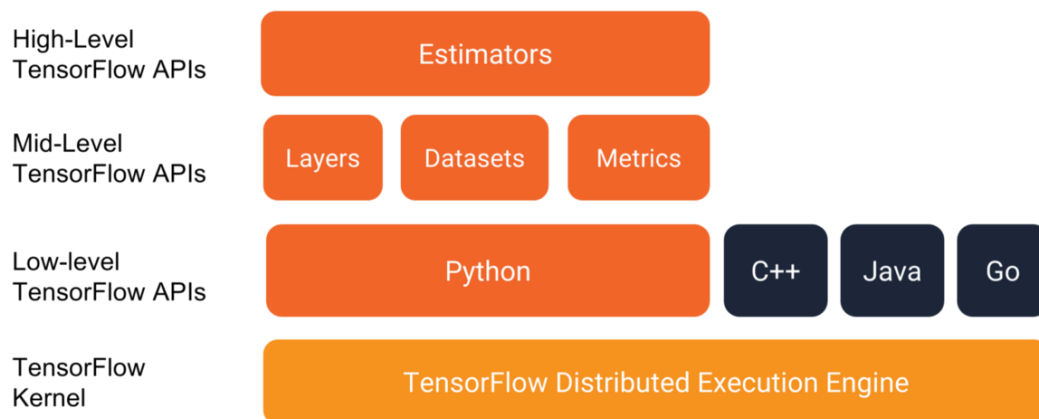
```
pip3 install tensorflow
```

TensorFlowHub

```
pip install tensorflow-hub
```

What is TensorFlow?

- Tensorflow is a library made by Google that makes math and machine learning engineering much easier. It abstracts many concepts such as back-propagation and neurons that require even more background knowledge of the underlying math of this machine learning!
- You can do a lot of different techniques, including CNNs! Below is a diagram of many of the things you can do with the TensorFlow API's!



Transfer Learning



- Transfer learning is retraining certain layers of an already general trained network so that you can use it for your specific use. **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.

- TensorFlow Hub lets you download already trained neural networks that you can do transfer learning on.

Advantages of transfer learning:

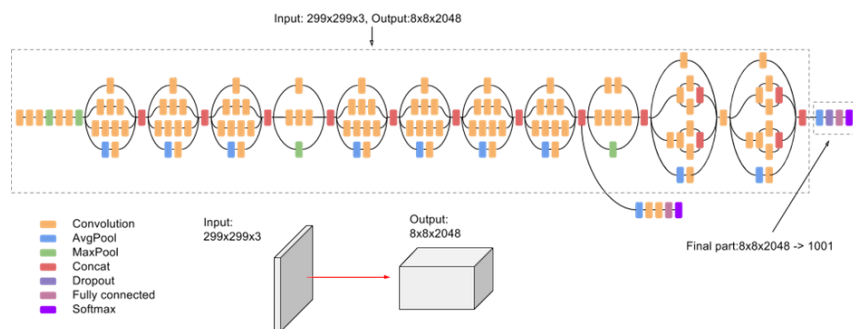
1. Takes less data to train
2. Takes a lot less processing time to train
3. **Less coding**

Disadvantages of transfer learning:

1. Less accurate than a custom CNN or fully training your own
2. No control over the inner layers of your CNN

Our Network: Inceptionv3

- 2nd place at ILSVRC 2015, which is an Image Recognition competition. Neural Networks like Inception v3 dominate at these type of competitions
- Made by a huge team of Google Machine Learning Engineers, so likely much better than anything that I or



Network Structure for Inception v3

retrain.py

- retrain.py is a python program that can easily retrain the final layer of any pretrained neural network that exists on TensorFlow Hub
- LINK:
https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py
- Usage:

```
python retrain.py --arg1 --arg2
```

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

retrain.py arguments

- -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
 L ...
 - 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 its weights from their previous values. A too high learning rate can diverge quickly in its loss graph and can in fact 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 its 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 image

label_image.py

After you have trained your model you can use `label_image.py` in order to actually classify images. It will return the probability for each class of what the network thinks the image could be.

LINK:

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/label_image/label_image.py

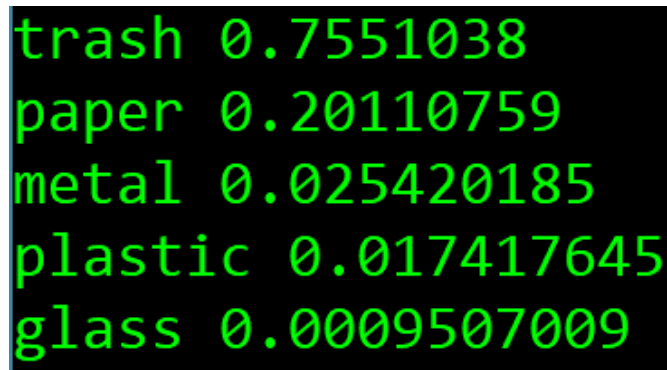
Usage:

```
python label_image.py --arg1 --arg2
```

label_image.py arguments

- `-h`
 - This argument lists all of the arguments that are covered in this documentation file and smaller

- **-image**
 - This is the location of the image that you want to classify
- **-graph (file)**
 - This is the output graph generated from retrain.py. The default location for the output graph 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
- **-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



```
trash 0.7551038
paper 0.20110759
metal 0.025420185
plastic 0.017417645
glass 0.0009507009
```

Output example of label_image.py on an image

References

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

Like

Be the first to like this.

[Edit](#)

[IEEE QP Wiki](#), [Powered by WordPress.com](#).