TensorFlow Image Classification Windows Machine

Instructions for Running / Training / Testing Image Classification with TensorFlow Using Inception on Windows 10

NB We are using tensorflow 1.13 for this module, tensorflow 2.0 beta is still on the test phase. Most model available are still on tensorflow 1_version some of the coding is being phased out. So until they update the models to 2.0 you rather stick to the highest version of version 1.

You can also run this system on your local machine without anaconda using other python GUI

1. **Set_up**

I assume you have installed anaconda or follow the following link

https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/

Create new environment by conda

If you are unwilling to create conda environment (maybe because of lazy), you can skip this section. However, I strongly recommend you to create this **for the convenience in the future**.

Run the command below:

```
conda create -n tf
```

If you do not have tensorflow installed

First, you need to change to the env you have just built by conda:

```
conda activate tf
```

Afterwards, type in the command to install TensorFlow preferably version 1.13.1, you need:

```
conda install tensorflow-gpu
```

It will install the latest version, so you will need to specify the version.

If you want to install a specific version of tensorflow-gpu or cpu veison, you can change the command like this:

```
conda install tensorflow-gpu=1.13.1  #if you want to install 1.13.1 version
conda install tensorflow  #if you want to install cpu version
```

After anaconda solve the environment, you just need to type in ‘y’ to confirm the installation.

Anaconda will automatically **install other libs and toolkits needed by tensorflow(e.g. CUDA, and cuDNN), so you have no need to worry about this.**

Type in `python` to enter the python environment.

```
import tensorflow as tf
tf.__version__
```

When you see the version of tensorflow, such as 1.13.1, you have successfully install it.

That’s all, Thank you.

Backdate to TensorFlow 1.13.1

With the current version of TensorFlow (2.0) there are some incompatibilities between TensorFlow 1.13.1 and models (more on models later) 2.0. To Remedy these incompatibilities it is necessary to backdate TensorFlow to 1.13.1.

Check your TensorFlow version:

```
pip list
```

If you have not installed pip,

```
Conda install pip
```

If not 1.13.1, open an ***administrative*** command prompt and enter:

```
pip uninstall tensorflow
```
(wait until the uninstall is complete)

(if using TensorFlow CPU version)
```
pip install --upgrade tensorflow==1.13.1
```

(if using TensorFlow GPU version)
```
pip install --upgrade tensorflow-gpu==1.13.1
```

1. Clone the repository containing this document

To clone the repository install git on windows

https://www.computerhope.com/issues/ch001927.htm

To clone

https://git-scm.com/docs/git-clone

Clone the repository containing this document:

In git:

cd "C:\Users\XYTB\Documents\ML\Github_loaded"

Clone the repository containing this document:

git clone

https://github.com/Mutekeri/TensorFlow_Image_Classification_Windows_Machine.git

to a location you'd like to work in, for example I'll use:

C:\Users\Michael\Documents\ML\Github_loaded

You can choose any directory you'd like. Going forward in this document we'll refer to this location as ***(repository_location)***

For jupyter notebook

Open a new folder in jupyter call it what you want
Open a new python file
Type in

C:\Users\DCFFS\Documents\ML\TensorFlow_Object_Detection_Windows_Machine

```
import zipfile as zf
files=zf.ZipFile(C:\Users\xvxvvx\Documents\ML\Github_loaded\TensorFlow_Image_Classification_Windows_Machine.zip,'r')
files.extractall(Image_Classification)
files.close()
```

Open the object detection folder, you will find another folder older
Go back
Rename the object detection to object_detetion1 by adding a 1
Go inside the folder, move the other Image_Classification folder just outside by removing /
Image_Classification_1 on the path, this is to prevent jupyter notebook to have errors when finding the information in your file.
There will be two folders object_detection1 and Image_Classification, delet the object_detection1 and open again
You will see the relevant files straight up, without getting in another folder

2. Download images

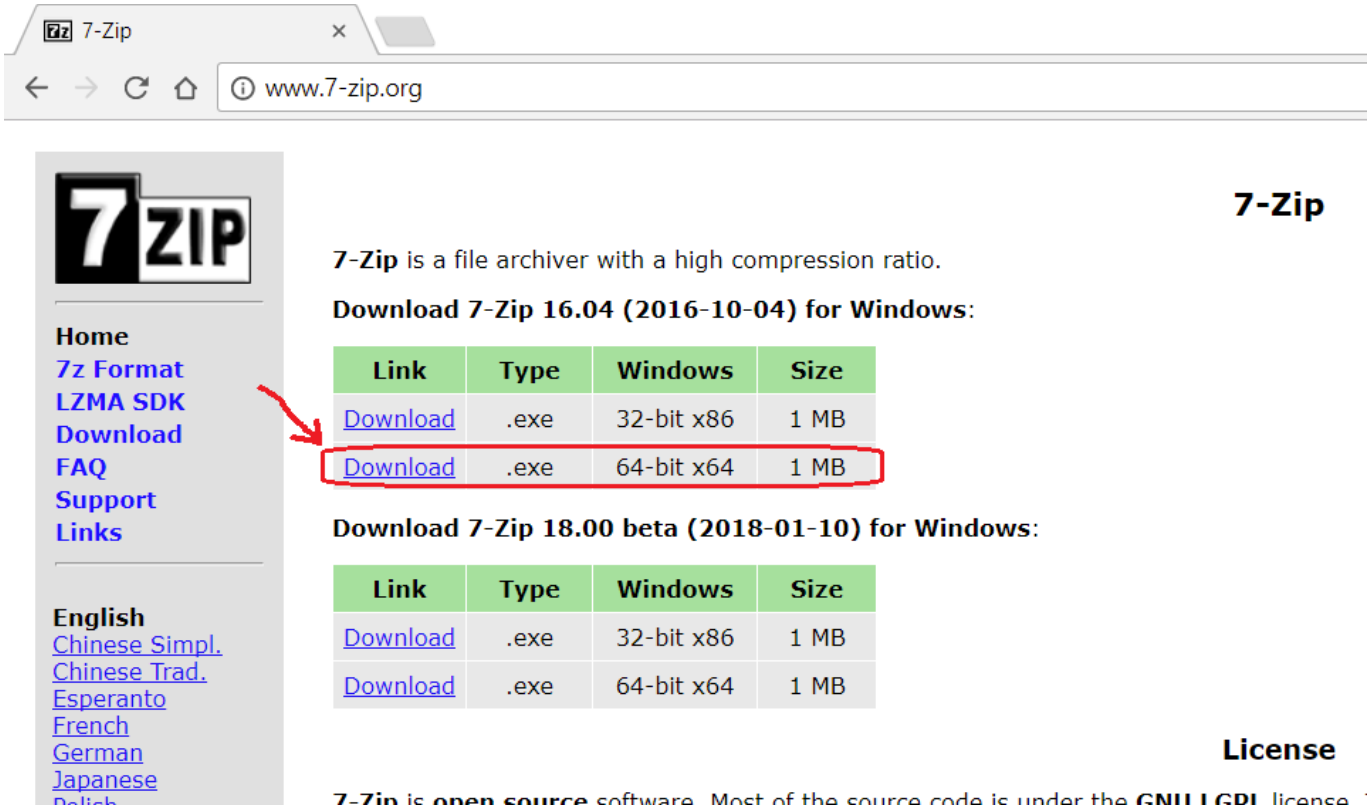
At this point you can choose to use either an image set of different flower types to classify provided by Google (follow step 2a), or you can use your own images if you prefer (follow step 2b).

2a. Download Google flower images

Download and save the flower_photos.tgz file from this link to ***(repository_location)***:

http://download.tensorflow.org/example_images/flower_photos.tgz

Download and install 7-Zip (yes, it's free) so you can unzip the .tgz file:



Once you've downloaded flower_photos.tgz to C:\tf_files, right-click on it and choose "7-Zip" -> "Extract Here". If "7-Zip" does not appear as a right-click option after installing 7-Zip, try rebooting and try your right-click menu again. Verify this step unzips the flower_photos.tgz into a single "flower_photos.tar" file.

Next, right-click on the "flower_photos.tar" file and choose "7-Zip" -> "Extract Here" again, verify this unzips the "flower_photos.tar" file into a "flower_photos" directory that contains 5 other directories, "daisy", "dandelion", "roses", "sunflowers", and "tulips", and also a LICENSE.txt file. Verify the 5 directories named after flower types each have 500-1,000 .jpg images.

Rename the "flower_photos" directory to "training_images".

2b. Download your own images

If you'd like to use your own images, create a directory "***(repository_location)***\training_images", then create a directory for each classification you'd like, then download at least 105 images of each type.

For example, if you'd like to classify road bikes vs mountain bikes, create the directories "***(repository_location)***\training_images\road_bikes" and "***(repository_location)***\training_images\mountain_bikes", then download at least 105 images of each and save them to the applicable directories.

The images can be different sizes, but should be roughly square, and not especially large or small (i.e. substantially bigger than 50 x 50 and substantially smaller than 4000 x 4000).

If you'd like to save time, you can use this collection of road and mountain bikes images I downloaded:
Google Drive:
https://drive.google.com/drive/folders/1ywyfiAEI0ql81gMy58UeamWvV7u9xGn9?usp=sharing
OneDrive:
https://1drv.ms/f/s!AoYpNs_C1pusgxvM3sOU5Wn9yJm5

Provided by github.com (MicrocontrollersAndMore) , The images are the same on both.

NB always use the zip instruction to load folders in you folder in jupyter notebook

The Train.ipynb script we will run in the next few steps may encounter an error is there are other files or directories inside each training directory that contains images, so take care to assure that only .jpg images are in each training images directory. In other words, for example, "***(repository_location)***\training_images\road_bikes" and "***(repository_location)***\training_images\mountain_bikes" should only contain .jpg images, not other file types or other directories.

5) Separate out some test images

Create a directory "***(repository_location)***\test_images". For each of the directories for the classifications in your ***(repository_location)***\training_images directory, choose at least 2 images and move (don't copy) them into ***(repository_location)***\test_images. Note that we are separating out the images we will use before training (the next step) so the images we test on will not have been used for training.

6) Run *Train.ipynb*

The file ***(repository_location)***\retrain.py is a refactored version of this file from the TensorFlow GitHub:

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/image_retraining/retrain.py

The refactorings are for improved readability and also to make using the command line not necessary.

Open ***(repository_location)***\retrain.py in your chosen Python editor and give it a quick skim. I moved the parameters that can be specified from the **if** __name__ == **'__main__'**: statement at the very bottom of the script to the "module level variables" section at the top. This should make things more user-friendly for Windows users who may not be used to using the command line. Note that there are more than 20 parameters that can be specified, which allows for a great variety of customization options, however the defaults that I've chosen should be good to start with.

retrain.py is more than 1,000 lines long so it's not necessary to read the entire file, but at a minimum it would be recommended to verify the TRAINING_IMAGES_DIR and TEST_IMAGES_DIR variables correctly matches your image directories locations. When you're ready, go ahead and run retrain.py.

Depending on your computer's horse power, the training process will probably take 20-30 minutes.

Take a quick glance at ***(repository_location)***, note that many files will have been downloaded/created from running retrain.py

7) Run *Test.ipynb*

In ***(repository_location)*** open test.py in your chosen Python editor. Verify the file and directory paths at the top are correct for your computer, then run the script. The script will open each image in TEST_IMAGE_DIR in an OpenCV window and show the classification results on standard out.

Done!!

If you'd like to make the accuracy better, the 2 general steps to accomplish this would be:
1) Use more training images. 500-1,000 may seem like a lot, but considering the variety of these images, more would be better. 10,000+ images would not be too many.
2) In retrain.py, set the how_many_training_steps parameter to something higher than 500. Google's default is 4000. This will make the training take longer, however.

The next tutorial will cover how to use the TensorFlow Object Detection API.