

Tensor Flow Classification Walk-through

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

Table of Contents:

- 1) Complete the previous walk-through.
 - 2) Complete the previous walk-through.
 - 3) Clone the repository containing this document
 - 4) Download images
 - 4a) Download Google flower images
 - 4b) Download your own images
 - 5) Separate out some test images
 - 6) Run [*retrain.py*](#)
 - 7) Run [*test.py*](#)
-

1) Complete the previous walk-through.

Complete this previous walk-through if you have not already:

[*\(put Link to video 1 here\)*](#)

https://github.com/MicrocontrollersAndMore/TensorFlow_Tut_1_Installation_and_First_Progs

2) Complete the previous walk-through.

Really, seriously, complete the above walk-through before continuing with this one:

[*\(put Link to video 1 here\)*](#)

https://github.com/MicrocontrollersAndMore/TensorFlow_Tut_1_Installation_and_First_Progs

This walk-through won't work if you haven't performed the steps in the previous walk-through. Go through the first walk-through before continuing if you haven't already.

3) Clone the repository containing this document

Clone the repository containing this document:

https://github.com/MicrocontrollersAndMore/TensorFlow_Tut_2_Classification_Walk-through

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

C:\Users\cdahms\Documents\TensorFlow_Tut_2_Classification_Walk-through

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

4) Download images

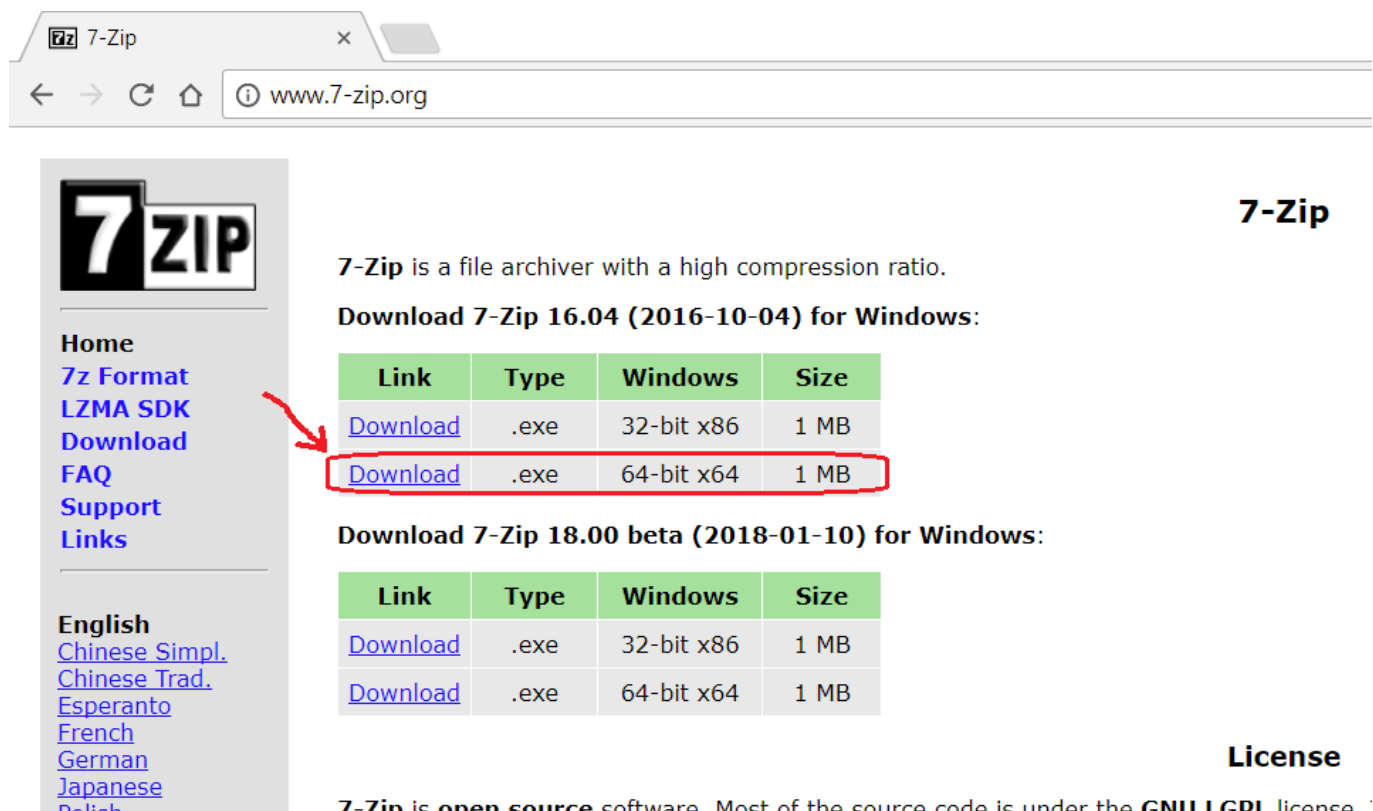
At this point you can choose to use either an image set of different flower types to classify provided by Google (follow step 4a), or you can use your own images if you prefer (follow step 4b).

4a) 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:



The screenshot shows the 7-Zip website. On the left is a sidebar with navigation links: Home, 7z Format, LZMA SDK, Download, FAQ, Support, and Links. Below these are language options: English, Chinese Simpl., Chinese Trad., Esperanto, French, German, Japanese, and Polish. The main content area has the 7-Zip logo and a description: "7-Zip is a file archiver with a high compression ratio." It then lists two download sections for Windows. The first section is for "Download 7-Zip 16.04 (2016-10-04) for Windows:" and contains a table with two rows. The second row, for the 64-bit x64 version, is highlighted with a red box and a red arrow points to its "Download" link. The second section is for "Download 7-Zip 18.00 beta (2018-01-10) for Windows:" and contains a similar table with two rows. At the bottom right, there is a "License" section.

Link	Type	Windows	Size
Download	.exe	32-bit x86	1 MB
Download	.exe	64-bit x64	1 MB

Link	Type	Windows	Size
Download	.exe	32-bit x86	1 MB
Download	.exe	64-bit x64	1 MB

License

7-Zip is open source software. Most of the source code is under the GNU LGPL license.

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”.

4b) 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

I’m providing links to both my Google Drive and my OneDrive with these images so if one or the other ever goes missing the other will still be available. The images are the same on both.

The retrain.py script we will run in the next few steps may encounter an error if 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 *retrain.py*

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 `IMAGE_DIR` variable correctly matches your image directory location. When you’re ready, go ahead and run it 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.py](#)

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.

You will notice there is a `ToDo` in test.py currently. test.py currently opens each test image twice, once in OpenCV NumPy array format, and again in TensorFlow format. I haven’t yet worked out a way to convert from NumPy array image format to TensorFlow image format. I have posted where I’m stuck on this on Stack Overflow: <https://stackoverflow.com/questions/48727264/how-to-convert-numpy-array-image-to-tensorflow-image>
If anybody could work out what I’m missing and respond that would be great.

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.