Instructions on Building the Icon Detector

Built On Windows

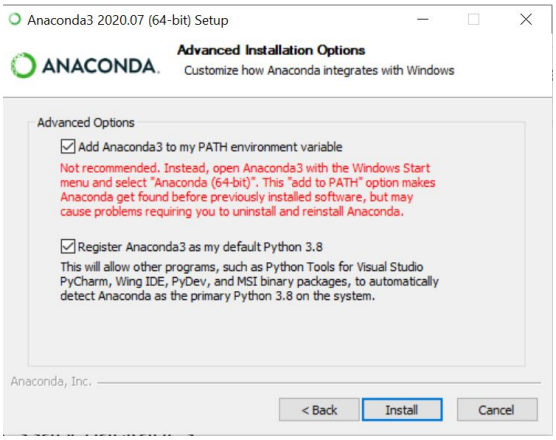
The instructions are tested on Windows 10 Home.

Environment Setup

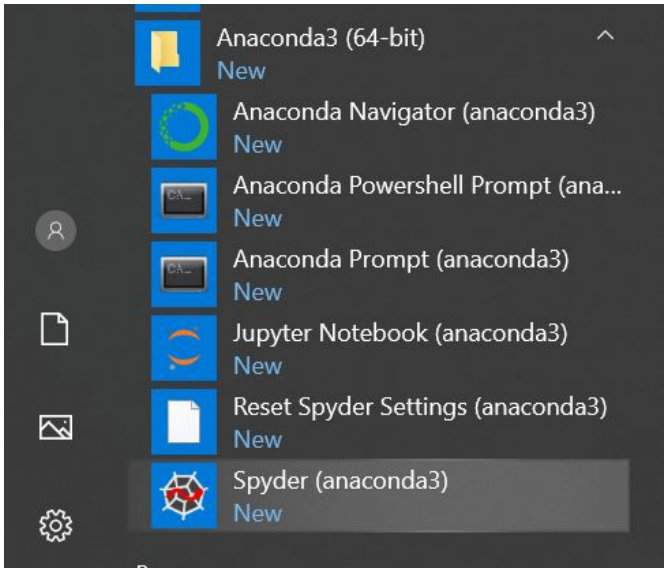
1. Install python packages

I prefer to install Anaconda as it is easy to install and build virtual python environments. We can download Anaconda3 Windows at https://www.anaconda.com/products/individual.

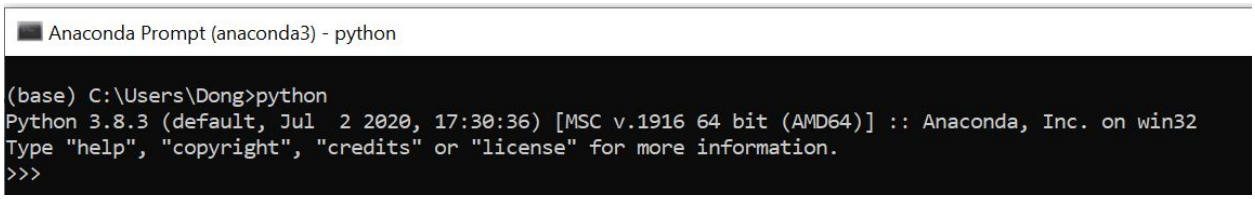
After downloading, we can install it and keep other options default while choosing to add Anaconda3 to your PATH.



After installation, you can initialize Anaconda3 at the start window. Open the **Anaconda prompt** and check the version of python.



You will see the output like this.



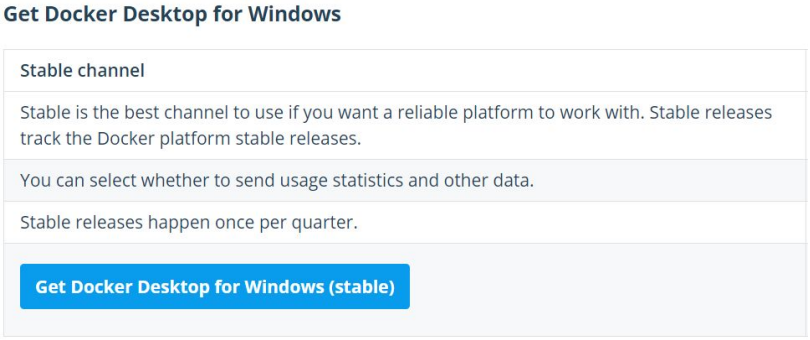
Congratulations! You can go to the second step.

2. Install Docker on Windows

As I am using Windows 10 Home, thus I installed Docker at the page https://docs.docker.com/docker-for-windows/install-windows-home/. You may use other

versions of Windows. You can go to the page: https://docs.docker.com/docker-for-windows/install/.

I installed the stable version of Docker.



After downloading, we can start to install Docker. You may face issues like,

Please update your Windows to the newest version. You can check the build of your Windows in the system. Please also check the solutions there https://docs.microsoft.com/en-us/windows/wsl/install-win10.

After downloading, we can install Docker following the instructions. After installing, we will run the command below to download the icon detector from the docker hub. This action is only required for one time.

```
docker pull dongchen93/icon-detector:v0
```

Then you can run the command below to run the server and keep the window (you can minimize it).

```
docker run -p 8500:8500 -t dongchen93/icon-detector:v0 &
```

```
2020-07-28 05:54:47.675620: I tensorflow_serving/model_servers/server.cc:355] Running gRPC ModelServer at 0.0.0.0:8500 .
..
[warn] getaddrinfo: address family for nodename not supported
[evhttp_server.cc : 238] NET_LOG: Entering the event loop ...
2020-07-28 05:54:47.693519: I tensorflow_serving/model_servers/server.cc:375] Exporting HTTP/REST API at:localhost:8501
...
```

3. Install necessary packages

As we are using tensorflow 1.14.0, thus we need to use python 3.5. So we can build a virtual python environment by the command and select ***yes***. You need to run it on the Anaconda Prompt window.

```
conda create -n py3.5 python=3.5
```

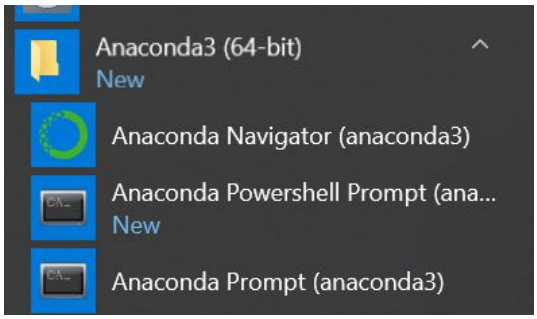
```
(base) C:\Users\Dong>conda create -n py3.5 python=3.5
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Dong\anaconda3\envs\py3.5
```

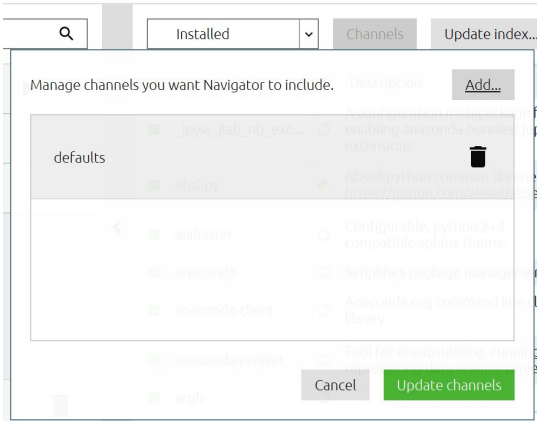
You may face problems like.

```
(base) C:\Users\Dong>conda create -n py3.5 python=3.5
Collecting package metadata (current_repodata.json): failed

UnavailableInvalidChannel: The channel is not accessible or is invalid.
  channel name: simple
  channel url: https://pypi.python.org/simple
  error code: 404

You will need to adjust your conda configuration to proceed.
Use `conda config --show channels` to view your configuration's current state,
and use `conda config --show-sources` to view config file locations.
```

You can solve it by

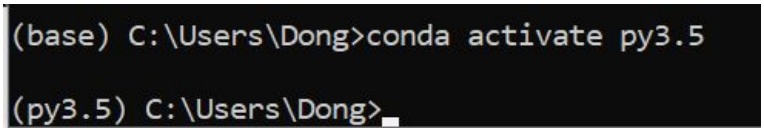
1. starting Anaconda Navigator,
2. clicking 'Environments',
3. clicking button 'Channels' and
4. removing the channel refering to my Downloads folder (only keeping 'defaults').





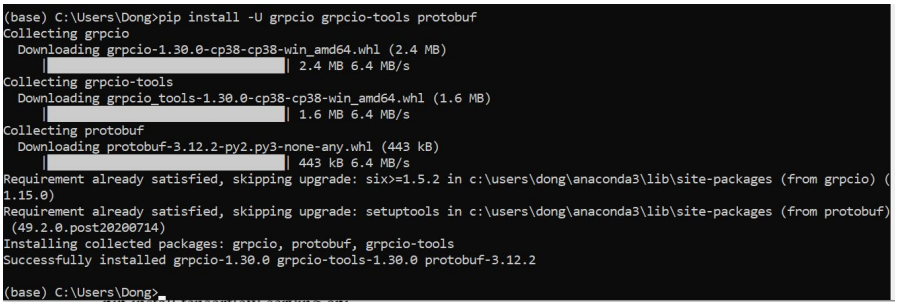
After you created the virtual environment, you can run it by the command.

conda activate py3.5



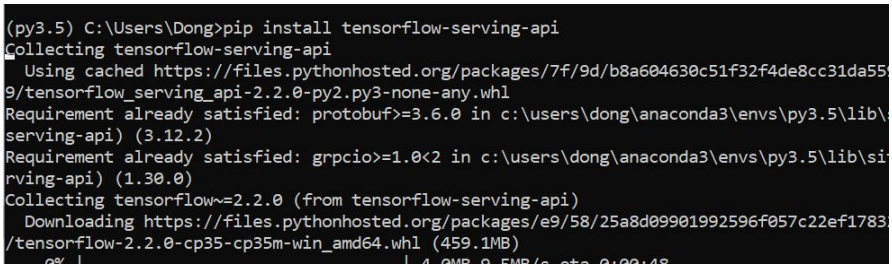
Now let's start to build the environment and install packages.

pip install -U grpcio grpcio-tools protobuf



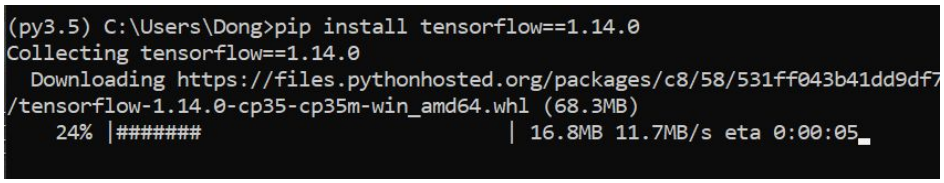
Install the Tensorflow server API by command.

pip install tensorflow-serving-api



Run the tensorflow 1.14.0 instead and install opencv-python.

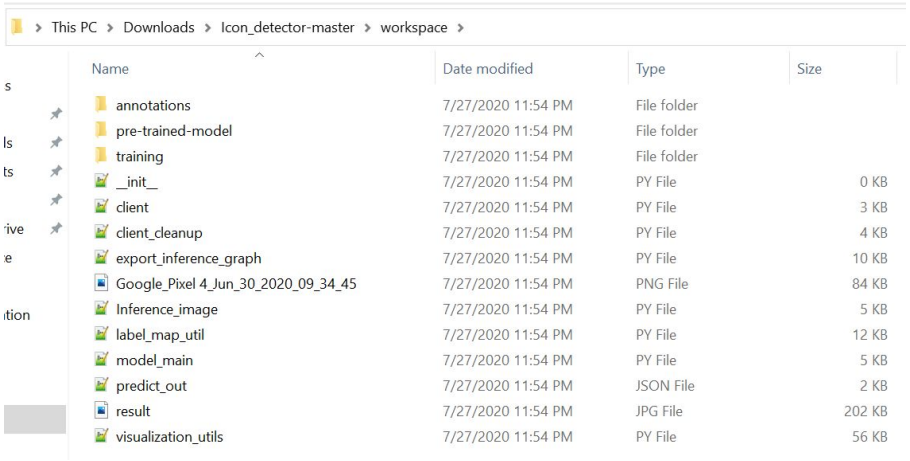
pip install tensorflow==1.14.0



pip install opencv-python

4. Download the Client Files

Now let's download the client files at https://github.com/Derekabc/Icon_detector and extract it.



Go to the client files with command.

cd C:\Users\Dong\Downloads\Icon_detector-master\workspace

```
(py3.5) C:\Users\Dong>cd C:\Users\Dong\Downloads\Icon_detector-master\workspace

(py3.5) C:\Users\Dong\Downloads\Icon_detector-master\workspace>
```

Run the client_cleanup.py file and get the detection output. We may need to run the clients several times to warm up the server. The running time varies depending on the setting of the PC. It takes about 0.3 - 0.5 s on the tested computer.

python client_cleanup.py

main.py | trainer.py | client.py | client_cleanup.py | predict_out.json


```
{"message": [58.10151144862175, 156.56674683094025, 371.7417937517166, 460.06588876247406, 95.0],
"star": [159.02658462524414, 231.9161832332611, 1761.6584515571594, 1848.145945072174, 86.0],
"n1": [156.74740970134735, 238.18275153636932, 1213.547773361206, 1365.589485168457, 98.0],
"n8": [498.0670952796936, 580.7721304893494, 1570.8418822288513, 1703.0801796913147, 100.0],
"n4": [158.22786033153534, 235.89293897151947, 1389.6539998054504, 1537.279007434845, 96.0],
"n0": [499.99252438545227, 582.2911405563354, 1738.9072608947754, 1884.7041606903076, 99.0],
"n7": [148.32271993160248, 241.40191733837128, 1564.458725452423, 1711.5314412117004, 99.0],
"n2": [497.63025999069214, 579.6884107589722, 1210.7440495491028, 1363.618004322052, 99.0],
"n3": [838.1529593467712, 923.4129595756531, 1210.4867935180664, 1362.169189453125, 98.0],
"n9": [834.377224445343, 937.9240536689758, 1568.1654953956604, 1717.7786993980408, 90.0],
"n5": [499.02602791786194, 582.7270746231079, 1389.4130516052246, 1544.938931465149, 99.0],
"n6": [838.8036417961121, 930.2708745002747, 1389.7930240631104, 1543.2105731964111, 99.0],
"sign": [848.6009573936462, 923.5059142112732, 1761.7122673988342, 1852.0103573799133, 79.0],
"call": [462.3737382888794, 618.2836389541626, 1947.8117966651917, 2108.3316779136658, 98.0]}
```


We can also change the input image by an input address with command like.

python client_cleanup.py --image "C:\\Users\\Dong\\Downloads\\Icon_detector-master\\Google_Pixel 4_Jun_30_2020_09_34_45.png"

5. Call the API on the cloud

In this method, we do not need to install docker locally and just edit the server address to 34.73.124.32 in client_cleanup.py by the command.

python client_cleanup.py --server '34.73.124.32:8500'

```
tf.app.flags.DEFINE_string('server', 'localhost:8500',  # 34.73.124.32   0.0.0.0   localhost
                           'PredictionService host:port')
```

It takes time to call the API on the cloud. It takes about 5s to complete a request. The time can be reduced by https://mux.com/blog/tuning-performance-of-tensorflow-serving-pipeline/.

Issues

Run docker server, you may come across issues like. You can release the ports by command.

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)

```
Dong@Dong-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker run -p 8500:8500 -t dongchen93/icon-detector:v0 &
[2] 15992

Dong@Dong-PC MINGW64 /c/Program Files/Docker Toolbox
$ C:\Program Files\Docker Toolbox\docker.exe: Error response from daemon: driver failed programming external connectivit
y on endpoint upbeat_feistel (7e096377128a61cca7cfb2bc7e83bfd1b59d624f1be67e0999d0e3710f5aba38): Bind for 0.0.0.0:8500 f
ailed: port is already allocated.

[2]+  Exit 125                docker run -p 8500:8500 -t dongchen93/icon-detector:v0

Dong@Dong-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker stop $(docker ps -a -q)
12dc6ac8cd15
eb79a54d625c
[1]+  Exit 137                docker run -p 8500:8500 -t dongchen93/icon-detector:v0

Dong@Dong-PC MINGW64 /c/Program Files/Docker Toolbox
$ docker rm $(docker ps -a -q)
12dc6ac8cd15
eb79a54d625c
```

You may come across issues when running client.py or client_cleanup.py

```
Traceback (most recent call last):
  File "client_cleanup.py", line 73, in <module>
    tf.app.run()
  File "C:\Users\Dong\anaconda3\envs\py3.5\lib\site-packages\tensorflow\python\platform\app.py", line 40, in run
    _run(main=main, argv=argv, flags_parser=_parse_flags_tolerate_undef)
  File "C:\Users\Dong\anaconda3\envs\py3.5\lib\site-packages\absl\app.py", line 299, in run
    _run_main(main, args)
  File "C:\Users\Dong\anaconda3\envs\py3.5\lib\site-packages\absl\app.py", line 250, in _run_main
    sys.exit(main(argv))
  File "client_cleanup.py", line 48, in main
    result = result_future.result()
  File "C:\Users\Dong\anaconda3\envs\py3.5\lib\site-packages\grpc\_channel.py", line 625, in result
    raise self
grpc._channel._MultiThreadedRendezvous: <_MultiThreadedRendezvous of RPC that terminated with:
        status = StatusCode.UNAVAILABLE
        details = "failed to connect to all addresses"
        debug_error_string = "{"created":"@1595915820.932000000","description":"Failed to pick subchannel","file":"src/c
ore/ext/filters/client_channel/client_channel.cc","file_line":3948,"referenced_errors":[{"created":"@1595915820.93200000
0","description":"failed to connect to all addresses","file":"src/core/ext/filters/client_channel/lb_policy/pick_first/p
ick_first.cc","file_line":394,"grpc_status":14}]}"
>
```

Try to use 0.0.0.0 instead of localhost.

```
tf.app.flags.DEFINE_string('server', 'localhost:8500',  # 34.73.124.32   0.0.0.0   localhost
                           'PredictionService host:port')
```