TensorFlow setup Documentation

Lyudmil Vladimirov

Jun 24, 2020

Contents:

This is a step-by-step tutorial/guide to setting up and using TensorFlow's Object Detection API to perform, namely, object detection in images/video.

The software tools which we shall use throughout this tutorial are listed in the table below:

Target Software versions	
OS	Windows, Linux
Python	3.8
TensorFlow	2.2
CUDA Toolkit	10.1
CuDNN	7.6.5

CHAPTER 1

Installation

1.1 TensorFlow Installation

1.1.1 Install TensorFlow pip package

- Open a new *Terminal* window
- Once open, type the following on the command line:

```
pip install --upgrade tensorflow
```

- Verify the install:

```
python -c "import tensorflow as tf;print(tf.reduce_sum(tf.random.
↪normal([1000, 1000])))"
```

Once the above is run, you should see a print-out similar to the one bellow:

```
2020-06-22 19:20:32.614181: W tensorflow/stream_executor/platform/default/
↪dso_loader.cc:55] Could not load dynamic library 'cudart64_101.dll';␣
↪dlerror: cudart64_101.dll not found
2020-06-22 19:20:32.620571: I tensorflow/stream_executor/cuda/cudart_stub.
↪cc:29] Ignore above cudart dlerror if you do not have a GPU set up on␣
↪your machine.
2020-06-22 19:20:35.027232: I tensorflow/stream_executor/platform/default/
↪dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
2020-06-22 19:20:35.060549: I tensorflow/core/common_runtime/gpu/gpu_
↪device.cc:1561] Found device 0 with properties:
pciBusID: 0000:02:00.0 name: GeForce GTX 1070 Ti computeCapability: 6.1
coreClock: 1.683GHz coreCount: 19 deviceMemorySize: 8.00GiB␣
↪deviceMemoryBandwidth: 238.66GiB/s
2020-06-22 19:20:35.074967: W tensorflow/stream_executor/platform/default/
↪dso_loader.cc:55] Could not load dynamic library 'cudart64_101.dll';␣
↪dlerror: cudart64_101.dll not found
```

(continues on next page)

(continued from previous page)

```
2020-06-22 19:20:35.084458: W tensorflow/stream_executor/platform/default/
↪dso_loader.cc:55] Could not load dynamic library 'cublas64_10.dll';␣
↪dlerror: cublas64_10.dll not found
2020-06-22 19:20:35.094112: W tensorflow/stream_executor/platform/default/
↪dso_loader.cc:55] Could not load dynamic library 'cufft64_10.dll';␣
↪dlerror: cufft64_10.dll not found
2020-06-22 19:20:35.103571: W tensorflow/stream_executor/platform/default/
↪dso_loader.cc:55] Could not load dynamic library 'curand64_10.dll';␣
↪dlerror: curand64_10.dll not found
2020-06-22 19:20:35.113102: W tensorflow/stream_executor/platform/default/
↪dso_loader.cc:55] Could not load dynamic library 'cusolver64_10.dll';␣
↪dlerror: cusolver64_10.dll not found
2020-06-22 19:20:35.123242: W tensorflow/stream_executor/platform/default/
↪dso_loader.cc:55] Could not load dynamic library 'cusparse64_10.dll';␣
↪dlerror: cusparse64_10.dll not found
2020-06-22 19:20:35.140987: I tensorflow/stream_executor/platform/default/
↪dso_loader.cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-06-22 19:20:35.146285: W tensorflow/core/common_runtime/gpu/gpu_
↪device.cc:1598] Cannot dlopen some GPU libraries. Please make sure the␣
↪missing libraries mentioned above are installed properly if you would␣
↪like to use GPU. Follow the guide at https://www.tensorflow.org/install/
↪gpu for how to download and setup the required libraries for your␣
↪platform.
Skipping registering GPU devices...
2020-06-22 19:20:35.162173: I tensorflow/core/platform/cpu_feature_guard.
↪cc:143] Your CPU supports instructions that this TensorFlow binary was␣
↪not compiled to use: AVX2
2020-06-22 19:20:35.178588: I tensorflow/compiler/xla/service/service.
↪cc:168] XLA service 0x15140db6390 initialized for platform Host (this␣
↪does not guarantee that XLA will be used). Devices:
2020-06-22 19:20:35.185082: I tensorflow/compiler/xla/service/service.
↪cc:176]   StreamExecutor device (0): Host, Default Version
2020-06-22 19:20:35.191117: I tensorflow/core/common_runtime/gpu/gpu_
↪device.cc:1102] Device interconnect StreamExecutor with strength 1 edge␣
↪matrix:
2020-06-22 19:20:35.196815: I tensorflow/core/common_runtime/gpu/gpu_
↪device.cc:1108]
tf.Tensor(1620.5817, shape=(), dtype=float32)
```

1.1.2 Install CUDA libraries (Optional)

Although using a GPU to run TensorFlow is not necessary, the computational gains are substantial. Therefore, if your machine is equipped with a compatible CUDA-enabled GPU, it is recommended to follow the steps listed below to install the relevant libraries necessary to enable TensorFlow to make use of your GPU.

By default, when TensorFlow is run it will attempt to register compatible GPU devices. If this fails, TensorFlow will resort to running on the platform's CPU. This can also be observed in the printout shown in the previous section, under the "Verify the install" bullet-point, where there are a number of messages which report missing library files (e.g. `Could not load dynamic library 'cudart64_101.dll'; dlerror: cudart64_101.dll not found`).

In order for TensorFlow to run on your GPU, the following requirements must be met:

Prerequisites
Nvidia GPU (GTX 650 or newer)
CUDA Toolkit v10.1
CuDNN 7.6.5

1.1.2.1 Install CUDA Toolkit

Windows

- Follow this link to download and install CUDA Toolkit 10.1
- Installation instructions can be found here

Linux

- Follow this link to download and install CUDA Toolkit 10.1 for your Linux distribution.
- Installation instructions can be found here

1.1.2.2 Install CUDNN

Windows

- Go to https://developer.nvidia.com/rdp/cudnn-download
- Create a user profile if needed and log in
- Select cuDNN v7.6.5 (Nov 5, 2019), for CUDA 10.1
- Download cuDNN v7.6.5 Library for Windows 10
- Extract the contents of the zip file (i.e. the folder named `cuda`) inside `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v10.1\`, where `<INSTALL_PATH>` points to the installation directory specified during the installation of the CUDA Toolkit. By default `<INSTALL_PATH>` = `C:\Program Files`.

Linux

- Go to https://developer.nvidia.com/rdp/cudnn-download
- Create a user profile if needed and log in
- Select cuDNN v7.6.5 (Nov 5, 2019), for CUDA 10.1
- Download cuDNN v7.6.5 Library for Linux
- Follow the instructions under Section 2.3.1 of the CuDNN Installation Guide to install CuDNN.

1.1.2.3 Environment Setup

Windows

- Go to *Start* and Search "environment variables"
- Click "Edit the system environment variables". This should open the "System Properties" window
- In the opened window, click the "Environment Variables..." button to open the "Environment Variables" window.
- Under "System variables", search for and click on the `Path` system variable, then click "Edit..."

- Add the following paths, then click "OK" to save the changes:
  - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v10.1\bin`
  - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v10.1\libnvvp`
  - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v10.1\extras\CUPTI\libx64`
  - `<INSTALL_PATH>\NVIDIA GPU Computing Toolkit\CUDA\v10.1\cuda\bin`

Linux

As per Section 7.1.1 of the CUDA Installation Guide for Linux, append the following lines to `~/.bashrc`:

```
# CUDA related exports
export PATH=/usr/local/cuda-10.1/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda-10.1/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_
↪PATH}}
```

1.1.2.4 Verify the install

Important: A new terminal window must be opened for the changes to the Environmental variables to take effect!!

As before, run the following command in a new *Terminal* window:

```
python -c "import tensorflow as tf;print(tf.reduce_sum(tf.random.normal([1000,␣
↪1000])))"
```

Once the above is run, you should see a print-out similar to the one bellow:

```
2020-06-22 20:24:31.355541: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cudart64_101.dll
2020-06-22 20:24:33.650692: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library nvcuda.dll
2020-06-22 20:24:33.686846: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561]␣
↪Found device 0 with properties:
pciBusID: 0000:02:00.0 name: GeForce GTX 1070 Ti computeCapability: 6.1
coreClock: 1.683GHz coreCount: 19 deviceMemorySize: 8.00GiB deviceMemoryBandwidth:␣
↪238.66GiB/s
2020-06-22 20:24:33.697234: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cudart64_101.dll
2020-06-22 20:24:33.747540: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cublas64_10.dll
2020-06-22 20:24:33.787573: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cufft64_10.dll
2020-06-22 20:24:33.810063: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library curand64_10.dll
2020-06-22 20:24:33.841474: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cusolver64_10.dll
2020-06-22 20:24:33.862787: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cusparse64_10.dll
2020-06-22 20:24:33.907318: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-06-22 20:24:33.913612: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1703]␣
↪Adding visible gpu devices: 0
2020-06-22 20:24:33.918093: I tensorflow/core/platform/cpu_feature_guard.cc:143] Your␣
↪CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
```

(continues on next page)

(continued from previous page)

```
2020-06-22 20:24:33.932784: I tensorflow/compiler/xla/service/service.cc:168] XLA␣
↪service 0x2382acc1c40 initialized for platform Host (this does not guarantee that␣
↪XLA will be used). Devices:
2020-06-22 20:24:33.939473: I tensorflow/compiler/xla/service/service.cc:176]  ␣
↪StreamExecutor device (0): Host, Default Version
2020-06-22 20:24:33.944570: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561]␣
↪Found device 0 with properties:
pciBusID: 0000:02:00.0 name: GeForce GTX 1070 Ti computeCapability: 6.1
coreClock: 1.683GHz coreCount: 19 deviceMemorySize: 8.00GiB deviceMemoryBandwidth:␣
↪238.66GiB/s
2020-06-22 20:24:33.953910: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cudart64_101.dll
2020-06-22 20:24:33.958772: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cublas64_10.dll
2020-06-22 20:24:33.963656: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cufft64_10.dll
2020-06-22 20:24:33.968210: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library curand64_10.dll
2020-06-22 20:24:33.973389: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cusolver64_10.dll
2020-06-22 20:24:33.978058: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cusparse64_10.dll
2020-06-22 20:24:33.983547: I tensorflow/stream_executor/platform/default/dso_loader.
↪cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-06-22 20:24:33.990380: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1703]␣
↪Adding visible gpu devices: 0
2020-06-22 20:24:35.338596: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102]␣
↪Device interconnect StreamExecutor with strength 1 edge matrix:
2020-06-22 20:24:35.344643: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108] ␣
↪     0
2020-06-22 20:24:35.348795: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1121]␣
↪0:   N
2020-06-22 20:24:35.353853: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1247]␣
↪Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 6284␣
↪MB memory) -> physical GPU (device: 0, name: GeForce GTX 1070 Ti, pci bus id:␣
↪0000:02:00.0, compute capability: 6.1)
2020-06-22 20:24:35.369758: I tensorflow/compiler/xla/service/service.cc:168] XLA␣
↪service 0x2384aa9f820 initialized for platform CUDA (this does not guarantee that␣
↪XLA will be used). Devices:
2020-06-22 20:24:35.376320: I tensorflow/compiler/xla/service/service.cc:176]  ␣
↪StreamExecutor device (0): GeForce GTX 1070 Ti, Compute Capability 6.1
tf.Tensor(122.478485, shape=(), dtype=float32)
```

Notice from the lines highlighted above that the library files are now "Successfully opened" and a debugging message is presented to confirm that TensorFlow has successfully "Created TensorFlow device".

1.1.2.5 Update your GPU drivers (Optional)

If during the installation of the CUDA Toolkit (see *Install CUDA Toolkit*) you selected the *Express Installation* option, then your GPU drivers will have been overwritten by those that come bundled with the CUDA toolkit. These drivers are typically NOT the latest drivers and, thus, you may wish to updte your drivers.

- Go to http://www.nvidia.com/Download/index.aspx
- Select your GPU version to download
- Install the driver for your chosen OS

1.2 TensorFlow Models Installation

1.2.1 Downloading the TensorFlow Models

- Create a new folder under a path of your choice and name it `TensorFlow`. (e.g. `C:\Users\sglvladi\Documents\TensorFlow`).
- From your *Terminal* `cd` into the `TensorFlow` directory.
- To download the models you can either use Git to clone the TensorFlow Models repository inside the `TensorFlow` folder, or you can simply download it as a ZIP and extract its contents inside the `TensorFlow` folder. To keep things consistent, in the latter case you will have to rename the extracted folder `models-master` to `models`.
- You should now have a single folder named `models` under your `TensorFlow` folder, which contains another 3 folders as such:

```
TensorFlow
└─ models
    ├── community
    ├── official
    ├── research
    └── ...
```

1.2.2 Protobuf Installation/Compilation

The Tensorflow Object Detection API uses Protobufs to configure model and training parameters. Before the framework can be used, the Protobuf libraries must be downloaded and compiled.

This should be done as follows:

- Head to the protoc releases page
- Download the latest `protoc-*-*.zip` release (e.g. `protoc-3.11.0-win64.zip` for 64-bit Windows)
- Extract the contents of the downloaded `protoc-*-*.zip` in a directory `<PATH_TO_PB>` of your choice (e.g. `C:\Program Files\Google Protobuf`)
- Extract the contents of the downloaded `protoc-*-*.zip`, inside `C:\Program Files\Google Protobuf`
- Add `<PATH_TO_PB>` to your `Path` environment variable (see *Environment Setup*)
- In a new *Terminal*[1], `cd` into `TensorFlow/models/research/` directory and run the following command:

```
# From within TensorFlow/models/research/
protoc object_detection/protos/*.proto --python_out=.
```

Important: If you are on Windows and using Protobuf 3.5 or later, the multi-file selection wildcard (i.e `*.proto`) may not work but you can do one of the following:

Windows Powershell

```
# From within TensorFlow/models/research/
Get-ChildItem object_detection/protos/*.proto | foreach {protoc "object_detection/
↪protos/$($_.Name)" --python_out=.}
```

[1] NOTE: You MUST open a new *Terminal* for the changes in the environment variables to take effect.

Command Prompt

```
# From within TensorFlow/models/research/
for /f %i in ('dir /b object_detection\protos\*.proto') do protoc object_
↪detection\protos\%i --python_out=.
```

1.2.3 Adding necessary Environment Variables

1. Install the `Tensorflow\models\research\object_detection` package by running the following from `Tensorflow\models\research`:

   ```
   # From within TensorFlow/models/research/
   pip install .
   ```

2. Add *research/slim* to your `PYTHONPATH`:

Windows

- Go to *Start* and Search "environment variables"
- Click "Edit the system environment variables". This should open the "System Properties" window
- In the opened window, click the "Environment Variables..." button to open the "Environment Variables" window.
- Under "System variables", search for and click on the `PYTHONPATH` system variable,
  - If it exists then click "Edit..." and add `<PATH_TO_TF>\TensorFlow\models\research\slim` to the list
  - If it doesn't already exist, then click "New...", under "Variable name" type `PYTHONPATH` and under "Variable value" enter `<PATH_TO_TF>\TensorFlow\models\research\slim`
- Then click "OK" to save the changes:

Linux

The Installation docs suggest that you either run, or add to `~/.bashrc` file, the following command, which adds these packages to your PYTHONPATH:

```
# From within tensorflow/models/research/
export PYTHONPATH=$PYTHONPATH:<PATH_TO_TF>/TensorFlow/models/research/slim
```

where, in both cases, `<PATH_TO_TF>` replaces the absolute path to your `TesnorFlow` folder. (e.g. `<PATH_TO_TF>` = `C:\Users\sglvladi\Documents` if `TensorFlow` resides within your `Documents` folder)

1.3 Test your Installation

- Open a new *Terminal* window
- Install `jupyter` (if not done so already) by running:

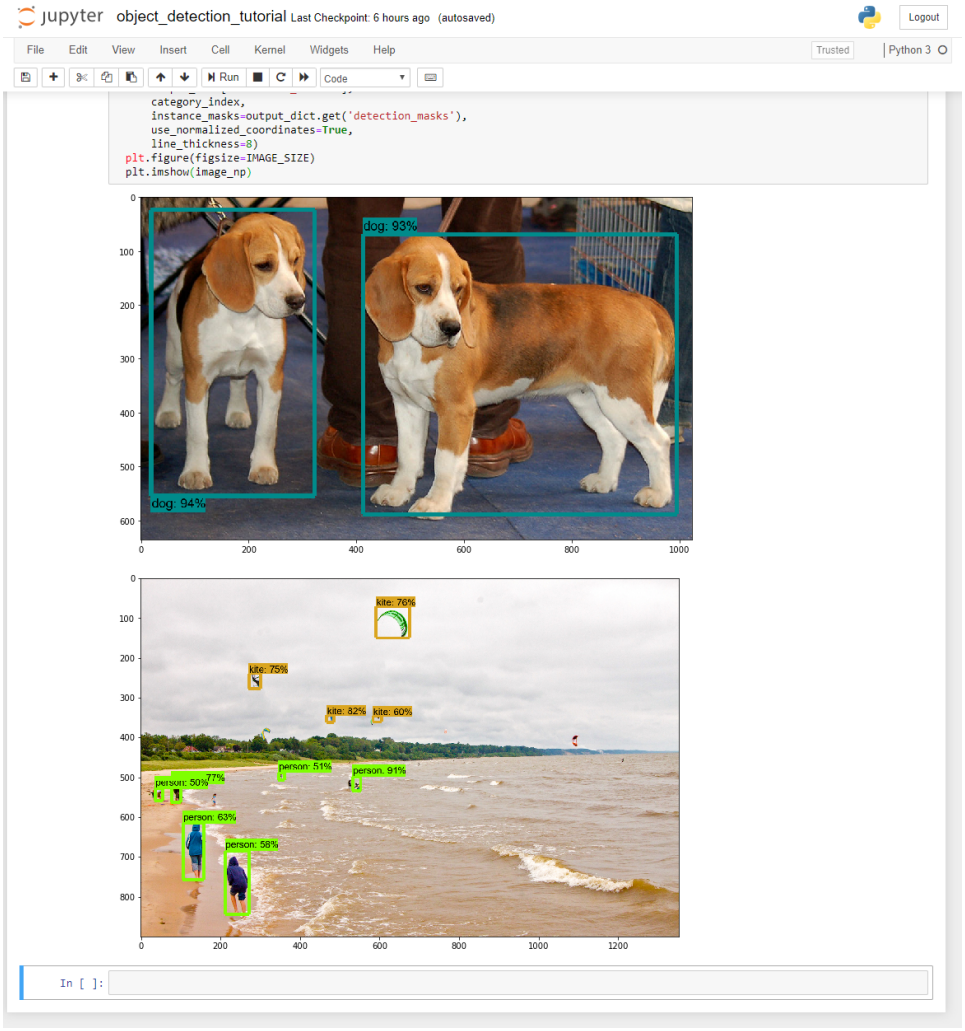
  ```
  pip install jupyter
  ```

- Download `this notebook` and paste it inside `TensorFlow\models\research\object_detection`.

- `cd` into `TensorFlow\models\research\object_detection` and run the following command:

```
# From within TensorFlow/models/research/object_detection
jupyter notebook
```

- This should start a new `jupyter notebook` server on your machine and you should be redirected to a new tab of your default browser.
- Once there, simply follow sentdex's Youtube video to ensure that everything is running smoothly.
- When done, your notebook should look similar to the image bellow:



CHAPTER 2

Indices and tables

- genindex
- modindex
- search