TENSORFLOW IN RED HAT VM

Objective	A guide for installation and operation of TensorFlow in Red Hat VM
Author	Andre Rosa
Date	05 OCT 2018
Version	1.0

Introduction

The objective of this document is to explain the steps taken to install and run TensorFlow in the Red Hat VM.

TensorFlow is a Python open-source machine learning library for research and production.

Development with TensorFlow demands Python and VS Code. There are different ways to install TensorFlow, we will use the Pip installation as explained below.

1 – Install Anaconda-Navigator (Python)

Install as explained in page: http://docs.anaconda.com/anaconda/install/linux/
ATTENTION: Select **Python 2.7** version.

At the end of installation you will be asked to install VSCode, if you already have VSCode installed there is not problem to answer 'yes' as the installer will check any prior installation.

ATTENTION: After installation close and open the terminal window again to run Anaconda. To run Anaconda just type anaconda-navigator in the terminal window.

2 – Check dependencies

Tensorflow demands the use of Python, Pip and VirtualEnv. For more detail check: https://www.tensorflow.org/install/pip?lang=python2

3 – Install TensorFlow with Conda

There are several ways to install TensorFlow the best way for those using Anaconda is with conda command line after Anaconda installation:

```
conda install -c conda-forge tensorflow
```

To install TensorFlow with Conda in a CUDA compatible system, GPU version, the command is (before choosing a version, look the comments bellow):

```
conda install -c conda-forge tensorflow-gpu
```

ATTENTION: ou should not install TensorFlow as explained in the TensorFlow home page (with Pip or Docker). Conda will install all dependencies as required by TensorFlow.

ATTENTION: For the Red Hat VM Conda will <u>install the TensorFlow CPU version</u>. The Red Hat VM does not seems to have CUDA. The tests for CUDA were made as described in: https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#pre-installation-actions

---

4 – Create and Activate Environment

Now we will create an Anaconda environment that will allow us to develop Python in VS Code. Tp proceed we will create an environment named 'tensorflow_env' with the command:

conda create -n tensorflow_env tensorflow

Answer "yes" to the qeustions and when done activate the environment with:

conda activate tensorflow_env

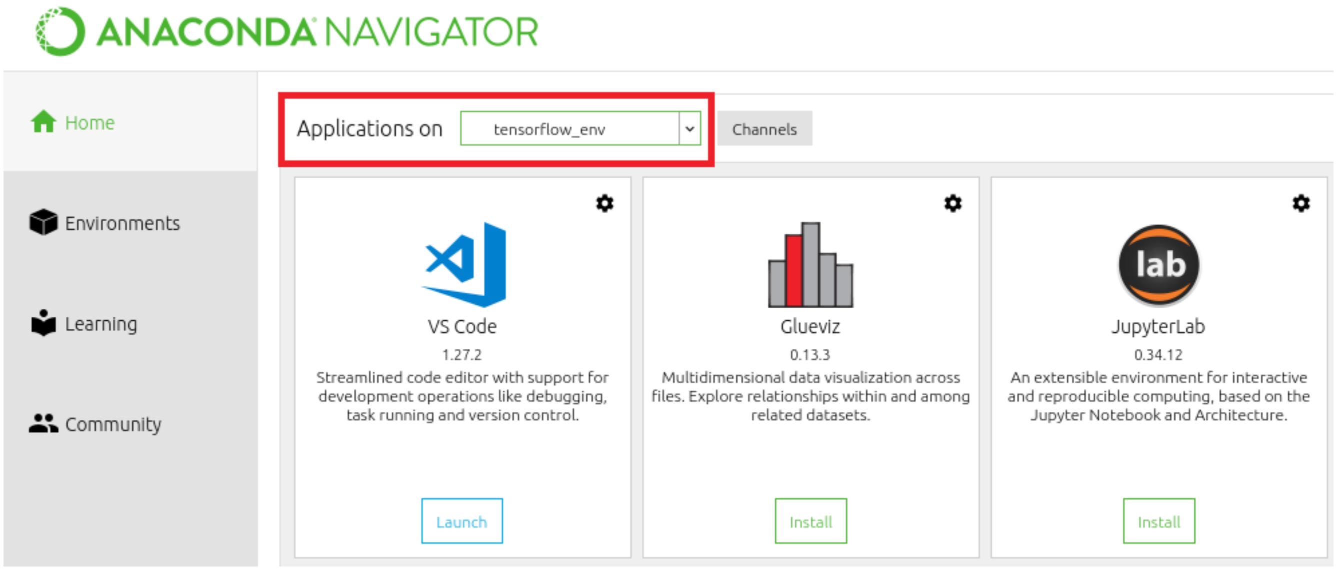
---

5 – Anaconda Navigator

Open a new terminal window and launch Anaconda with the command:

anaconda-navigator

In Anaconda-Navigator select the environment 'tensorflow_env' in "Application on" option, as shown in the picture bellow.



Streamlined code editor with support for development operations like debugging, task running and version control.

Multidimensional data visualization across files. Explore relationships within and among related datasets.

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

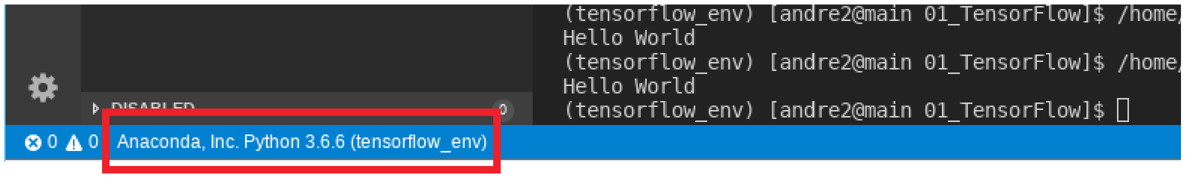
---

6 – Launch VS Code

From Anaconda launch VS Code. Three extensions will be already installed in VS Code to facilitate your Python development: Python, YAML and Anaconda.

If your VS Code asks for Lint installation proceed. Lint analyzes Python code for errors.

To be sure that everything is right check the bottom of the VS Code environment, you shall see a status bar like this:



```
(tensorflow_env) [andre2@main 01_TensorFlow]$ /home/
Hello World
(tensorflow_env) [andre2@main 01_TensorFlow]$ /home/
Hello World
(tensorflow_env) [andre2@main 01_TensorFlow]$ 
```

ATTENTION: Albeit it was installed the Python 2.7 version of Anaconda, somehow Python was updated it to 3.6.6. Right now both versions are installed in the VM. The version 3.6.6 was installed inside the tensorflow_env created on step 4 above. I am not going to investigate it further because the code is running fine.

---

7 – Test if TensorFlow is running

Open terminal window in VS Code in menu *View* > *Terminal*

Create a file and paste the following code:

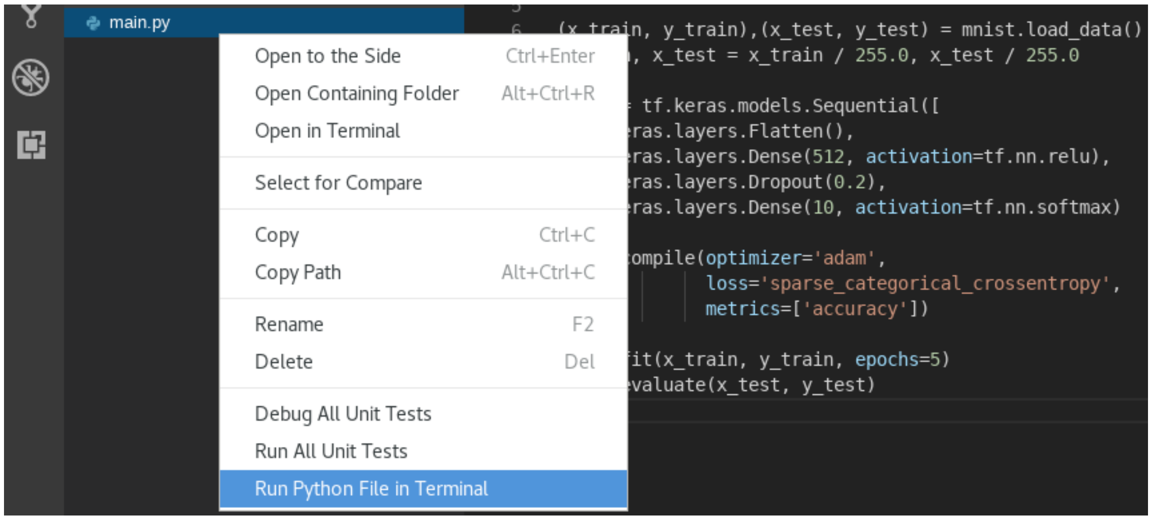
```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
             loss='sparse_categorical_crossentropy',
             metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Save and select with a ‘Right-Click’ the option ‘Run Python File in Terminal’.



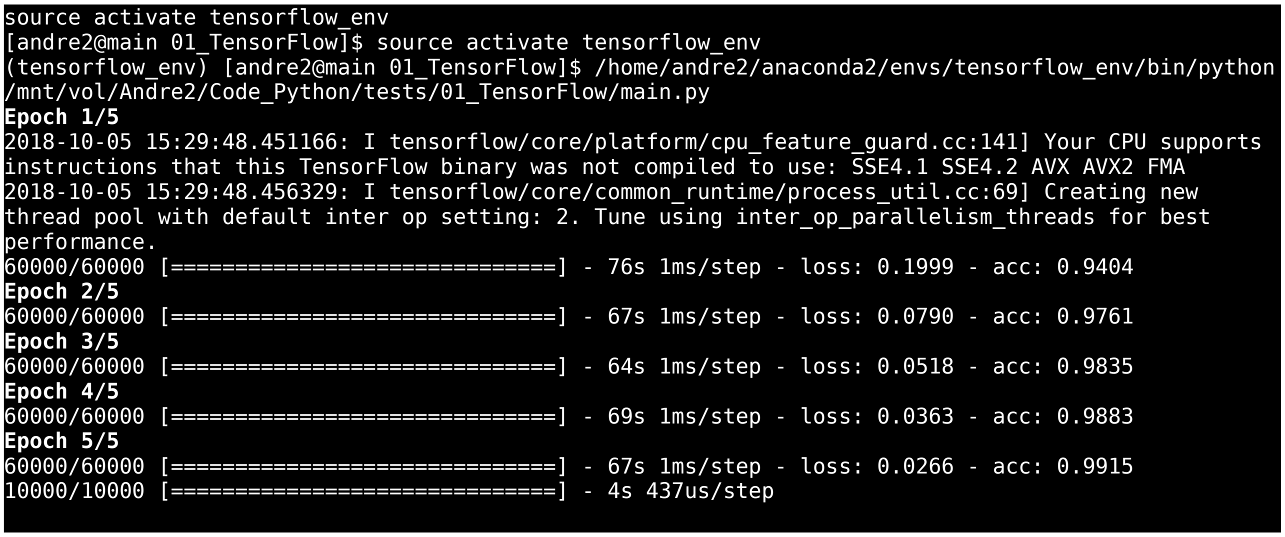
```
(x_train, y_train),(x_test, y_test) = mnist.load_data()
, x_test = x_train / 255.0, x_test / 255.0

tf.keras.models.Sequential([
ras.layers.Flatten(),
ras.layers.Dense(512, activation=tf.nn.relu),
ras.layers.Dropout(0.2),
ras.layers.Dense(10, activation=tf.nn.softmax)

ompile(optimizer='adam',
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy'])

it(x_train, y_train, epochs=5)
valuate(x_test, y_test)
```

Your code will output a series of tests of NN layers in the terminal area like this:



```
source activate tensorflow_env
[andre2@main 01_TensorFlow]$ source activate tensorflow_env
(tensorflow_env) [andre2@main 01_TensorFlow]$ /home/andre2/anaconda2/envs/tensorflow_env/bin/python
/mnt/vol/Andre2/Code_Python/tests/01_TensorFlow/main.py
Epoch 1/5
2018-10-05 15:29:48.451166: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports
instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
2018-10-05 15:29:48.456329: I tensorflow/core/common_runtime/process_util.cc:69] Creating new
thread pool with default inter op setting: 2. Tune using inter_op_parallelism_threads for best
performance.
60000/60000 [==============================] - 76s 1ms/step - loss: 0.1999 - acc: 0.9404
Epoch 2/5
60000/60000 [==============================] - 67s 1ms/step - loss: 0.0790 - acc: 0.9761
Epoch 3/5
60000/60000 [==============================] - 64s 1ms/step - loss: 0.0518 - acc: 0.9835
Epoch 4/5
60000/60000 [==============================] - 69s 1ms/step - loss: 0.0363 - acc: 0.9883
Epoch 5/5
60000/60000 [==============================] - 67s 1ms/step - loss: 0.0266 - acc: 0.9915
10000/10000 [==============================] - 4s 437us/step
```

Congratulations your TensorFlow installation with Anaconda and Python is successful.

---

8 – Using Jupyter Qt Console

Albeit you can run your code and get output with VS Code one interesting feature of Python is the capacity to generate visual output with libraries as mapplotlib. Qt Console is also part of the Anaconda Navigator and works as a terminal capable of generating colorful output unlike VS Code.

The code in the example below is found in GitHub and run in Qt Console with the command: ***run categorize.py***

https://github.com/Cadesh/TensorFlow/tree/master/ClothesCategorize)

Sources:

Anaconda Installation:
http://docs.anaconda.com/anaconda/install/linux/

TensorFlow Installation with Conda:
https://anaconda.org/conda-forge/tensorflow
https://www.anaconda.com/blog/developer-blog/tensorflow-in-anaconda/

TesorFlow Installation page:
https://www.tensorflow.org/install/pip

nVidia Testing CUDA;
https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html#pre-installation-actions

Python with VS Code
https://code.visualstudio.com/docs/languages/python

Versions Used:

- **Python** 2.7.5 from terminal with command: python -V
- **Python** 3.6.6 from tensorflow_env with terminal: python -V
- **Anaconda** 5.6
- **Anaconda-Navigator** 1.9.2
- **VS Code** 1.27.2
- **TensorFlow** 1.11.0 check with terminal: python -c 'import tensorflow as tf; print(tf.__version__)'
- **Qt Console** 4.3.1
