Lab 9 and 10: Data analytics in the Cloud

Introduction

In this lab, you will be learn how to perform a large scale data mining task such as Deep learning on the AWS and Google Cloud. If you are unable to finish it in Week 9, you can continue this lab in Week 10.

1 Data analytics in AWS

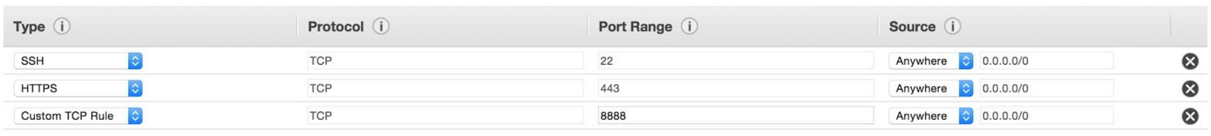
1. 1 Setup

1.1.1 AWS EC2 Instance

To set up your EC2 instance, follow steps 1 to 7 (Excluding step 3a, see below) according to the description of this link https://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learning-using-the-aws-deep-learning-ami/

Note:

- In step 3a, you have to select **m5.2xlarge** instead of **p3.2xlarge**.
- Make sure you assign the security group settings as shown in the screenshot.



Type	Protocol	Port Range	Source	
SSH	TCP	22	Anywhere	0.0.0.0/0
HTTPS	TCP	443	Anywhere	0.0.0.0/0
Custom TCP Rule	TCP	8888	Anywhere	0.0.0.0/0

- The connection process is similar what you did in **lab 5**. If you forget please go back to the document of lab 5 and follow the steps (according to your OS) to connect AWS AMI from command line/terminal

Once you are connected, at the top of terminal, you can see different commands to activate different conda environments but in this exercise you will be using **Python 3.6, Keras with Tensorflow backend**. To activate this, type

```
$ source activate tensorflow_p36
```

1.1.2 Jupyter Notebook

It is highly recommended to use Jupyter Notebook for deep learning tasks. In order to securely access Notebook remotely from a url, please follow the steps below.

1. Open the iPython Terminal (as using the command below) to get an encrypted password. You will need to use the password for logging into our

Jupyter Notebook Server. Please remember to copy and save the password you just entered and output of this command, which will be an encrypted password, something like "sha1..."

```
$ ipython
In [1]:from IPython.lib import passwd
In [2]:passwd()
```

and exit out of ipython terminal using "exit" command.

2. Now we are going to create the configuration profile for our Jupyter Notebook server.

   ```
   $ jupyter notebook --generate-config -y
   ```

3. The next thing is to create a self-signed certificate for accessing our Notebooks through HTTPS

   ```
   $ mkdir certs
   $ cd certs
   $ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.key -
   out mycert.pem
   ```

   This will ask some questions, please answer them to the best of your knowledge as some of them are required to successfully create the certificate.

4. Now, time to change the config settings of our server

   ```
   $ cd ~/.jupyter/
   $ vi jupyter_notebook_config.py
   ```

5. Add the following settings to the top of the file and leave the rest commented as it is.

   ```
   c = get_config()
   # Kernel config
   c.IPKernelApp.pylab = 'inline'  # if you want plotting support always in your
   notebook
   # Notebook config
   c.NotebookApp.certfile = u'/home/ubuntu/certs/mycert.pem' #location of your
   certificate file
   c.NotebookApp.keyfile = u'/home/ubuntu/certs/mycert.key' #location of your
   certificate key
   c.NotebookApp.ip = '*'
   ```

```
c.NotebookApp.open_browser = False  #so that the ipython notebook does not
opens up a browser by default
c.NotebookApp.password = u'[Replace with SHA you previously copied]'  #the
encrypted password we generated above
# It is a good idea to put it on a known, fixed port
c.NotebookApp.port = 8888
```

6. Changed the ownership of the /home folder and ~/.local/share/jupyter/ folder to current user running this command:

   ```
   $ cd ../..
   $ sudo chown -R $USER /home/
   $ sudo chown -R $USER ~/.local/share/jupyter/
   ```

7. We are almost done. Now, it is the time to start our Jupyter notebook server. For this, first I'll create a new folder which will store all my notebooks

   ```
   $ cd ~
   $ mkdir Notebooks
   $ cd Notebooks
   ```

8. Start my notebook server

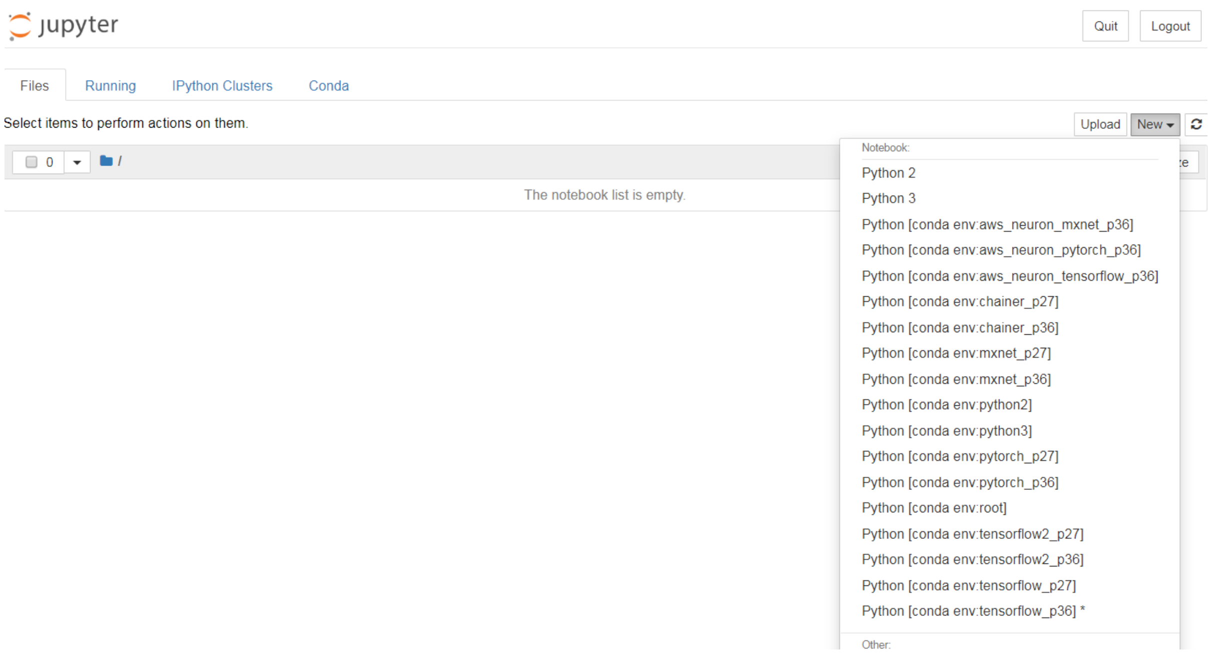
   ```
   $ jupyter notebook
   ```

9. And, that is all. Now you can access your Notebook from anywhere through your browser by going to https://[replace it with public DNS of the instance]:8888/

10. Proceed to website address (unsafe).

11. Login using the password you specified when you used the iPython Terminal to create an encrypted version of it and you are good to go.

12. On tree page, you can start a new notebook by clicking “Python [conda env:tensorflow_p36]” which is present in the menu when you click on the new tab on the top right side of the page.



Select items to perform actions on them.

The notebook list is empty.

Notebook:
Python 2
Python 3
Python [conda env:aws_neuron_mxnet_p36]
Python [conda env:aws_neuron_pytorch_p36]
Python [conda env:aws_neuron_tensorflow_p36]
Python [conda env:chainer_p27]
Python [conda env:chainer_p36]
Python [conda env:mxnet_p27]
Python [conda env:mxnet_p36]
Python [conda env:python2]
Python [conda env:python3]
Python [conda env:pytorch_p27]
Python [conda env:pytorch_p36]
Python [conda env:root]
Python [conda env:tensorflow2_p27]
Python [conda env:tensorflow2_p36]
Python [conda env:tensorflow_p27]
Python [conda env:tensorflow_p36] *
Other:

The first cell in a new notebook is always a code cell. Let's test it out with a classic hello world example. Type print('Hello World!') into the cell and click the run button ▶ Run in the toolbar above or press Ctrl + Enter.

To add more cells, click Insert and select Insert Cell Below to create a new code cell underneath your first or simply click + icon.

1. 2. Deep Learning

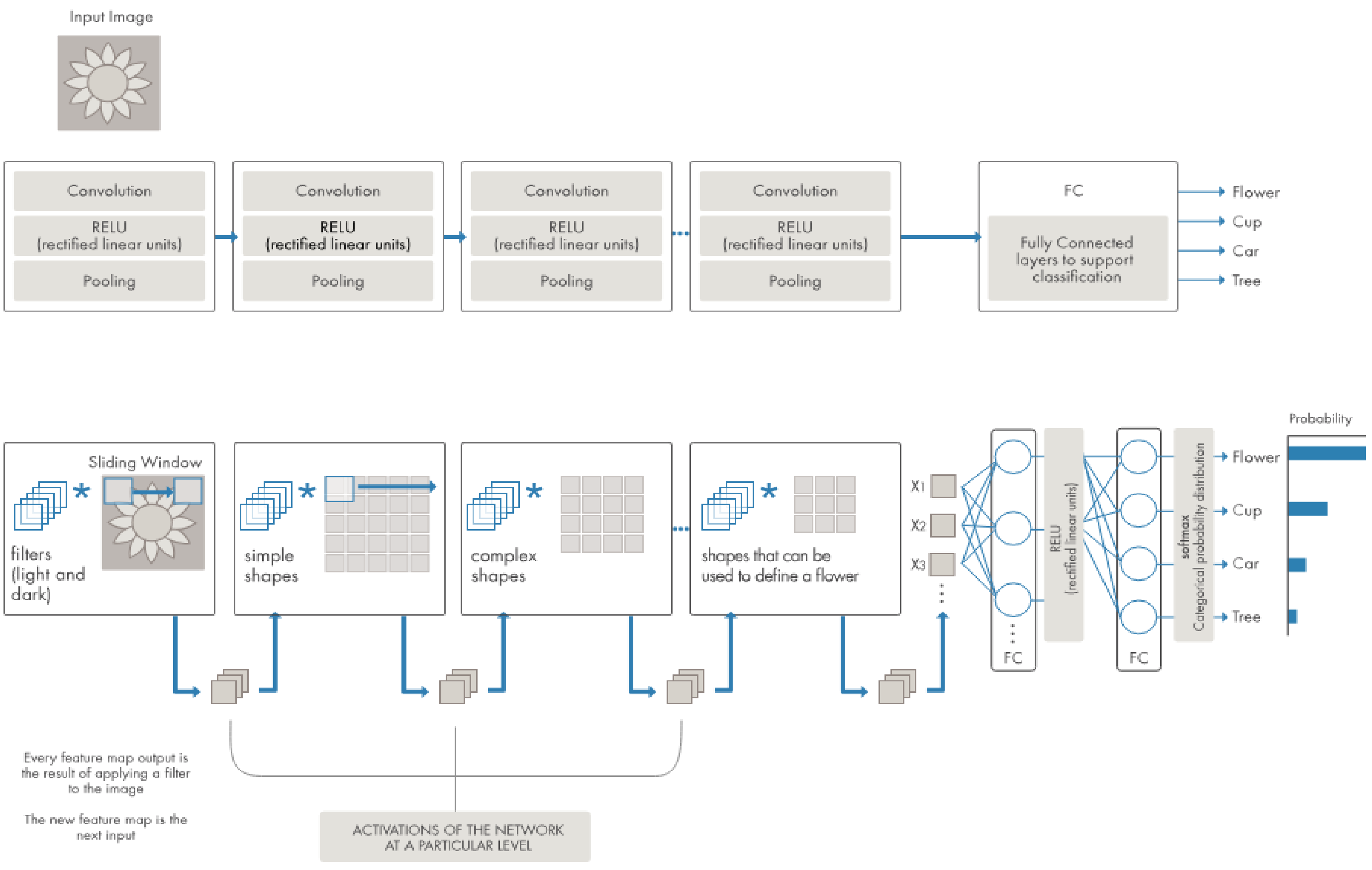
The data set we will be using here is the MNIST dataset. The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits (0 to 9). The database contains 60,000 training images and 10,000 testing images each of size 28x28.

This is a sample from MNIST dataset.

The task is to do image classification i.e predicting the digits in the images using Convolutional Neural Network (CNN). This will be designed using **Keras** with **Tensorflow** backend.

A CNN network is a combination of Convolution layer, pooling layer and fully connected layers. For further understanding, check this link http://cs231n.github.io/convolutional-networks/

Below is an example of CNN architecture and its working:



Input Image
Convolution
RELU (rectified linear units)
Pooling
Convolution
RELU (rectified linear units)
Pooling
Convolution
RELU (rectified linear units)
Pooling
Convolution
RELU (rectified linear units)
Pooling
FC
Fully Connected layers to support classification
Flower
Cup
Car
Tree
Sliding Window
filters (light and dark)
simple shapes
complex shapes
shapes that can be used to define a flower
X1
X2
X3
FC
RELU (rectified linear units)
FC
softmax
Categorical probability distribution
Probability
Flower
Cup
Car
Tree
Every feature map output is the result of applying a filter to the image
The new feature map is the next input
ACTIVATIONS OF THE NETWORK AT A PARTICULAR LEVEL

There are two stages involved in deep learning process – training (section **1.2.1**) and prediction (section **1.2.2**).

Deep Learning models, learn by examples. Therefore, the first step is to get the model trained on the images in the train set, in order to learn features from the images. CNNs learn to detect different features of an image using multiple hidden layers. Every hidden layer increases the complexity of the learned image features. For example, the first hidden layer could learn how to detect edges, and the last learns how to detect more complex shapes specifically catered to the shape of the object we are trying to recognize.

Secondly, the model is evaluated on unseen images in test data. A model makes classification decision based on the feature it learned during training. So, the accuracy of a model depends on the quantity and quality of the training set. Hence more the better. If model gives good results/accuracy for evaluation, then it can be used to predict classes for unseen data.

1.2.1 Training

Now it’s time to do deep learning. You can copy the code from model.py file available on Canvas and run in new code cell. The code will perform the following

1. Import python packages.
2. Download data from AWS s3.
3. Data Pre-processing.
4. Model Building, training, and evaluation.

The output shows the training progress and accuracy on the test model. Each epoch must have used around an average of 35 seconds, but this will be longer if you are using CPUs.

1.2.2 Prediction

At this point, you have a trained model, so lets do some prediction on unseen data. Copy the code from predict.py and run in new code cell. The small code chunk takes random images from the test set and visualizes it as well as the predicted categories for those images.

2 Google Colab

In last section, AWS CPUs were used to perform deep learning and it managed to run the code in a reasonable time of some seconds per epoch. But most of the times, the task can be very complex and requires more computational power i.e GPU, in order to make the training process faster. So this section will introduce you to the Google Colab.

Google Colab or "the Colaboratory" is a free cloud service hosted by Google to encourage Machine Learning and Artificial Intelligence research, where often the barrier to learning and success is the requirement of tremendous computational power.

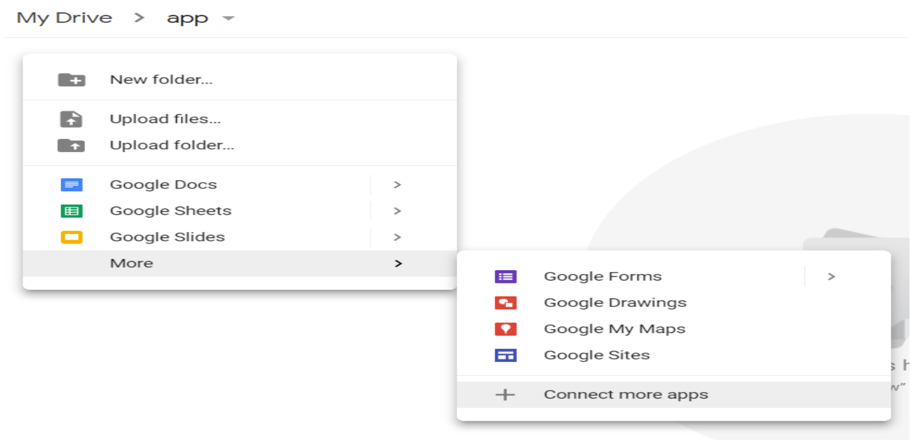
Benefits of Colab

- Much easier than AWS and other cloud services, in terms of setting up your account and environment.
- Supports Python 2.7 and Python 3.6 with free Tesla K80 GPU acceleration.
- All major Python libraries like TensorFlow, Scikit-learn, Matplotlib among many others are pre-installed and ready to be imported.
- Built on top of Jupyter Notebook.
- Collaboration feature (works with a team just like Google Docs): Google Colab allows developers to use and share Jupyter notebook amongst each other without having to download, install, or run anything other than a browser.
- Supports bash commands.
- Google Colab notebooks are stored on the drive.

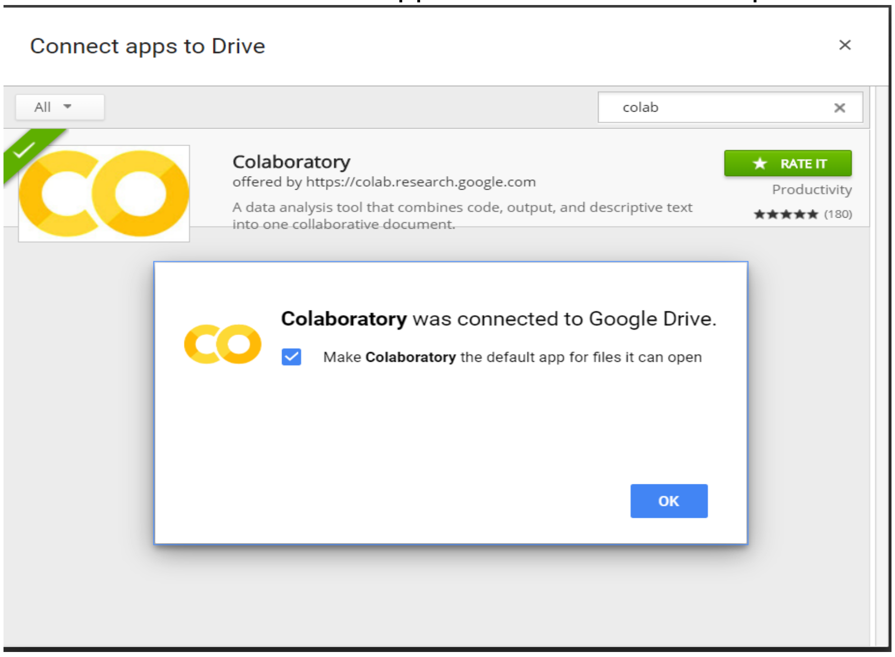
Detailed information about the service can be found on the faq page.

2.1 Setup

1. Login to your personal gmail account and open google drive.
2. Create a folder with name "app"and change directory to that folder.
3. Right click on the screen and look for "connect more apps" in "More" menu.



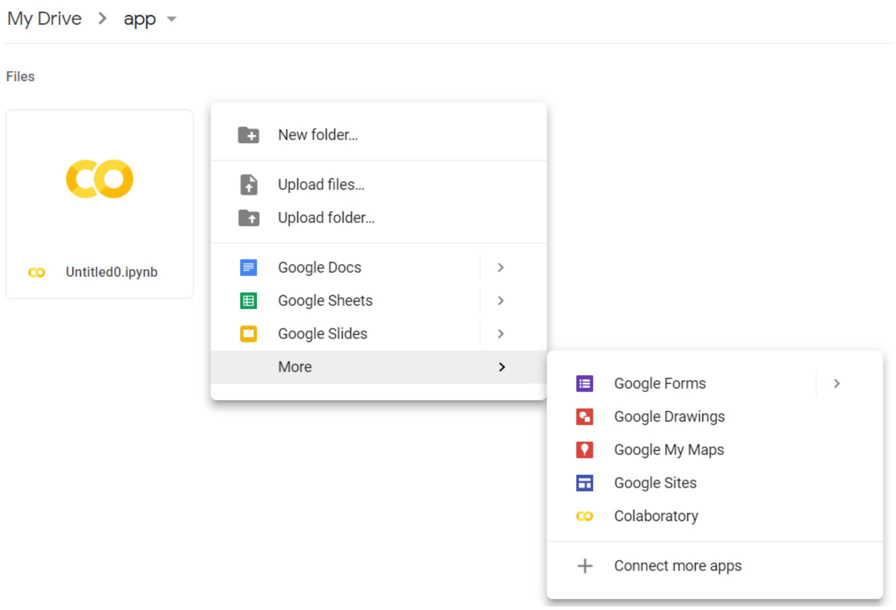
4. A window will open, there you can type “colab” in search field. Once the app appears in the list then click on “connect” button to install it. You may click on the check box to make it a default app for all the files it can open.


Connect apps to Drive
All
colab
Colaboratory
offered by https://colab.research.google.com
A data analysis tool that combines code, output, and descriptive text into one collaborative document.
RATE IT
Productivity
★★★★★ (180)
OK

Colaboratory was connected to Google Drive.

✓ Make **Colaboratory** the default app for files it can open

5. Now the Colaboratory option should appear in the “more” menu. Click on it to create a new Colab notebook.

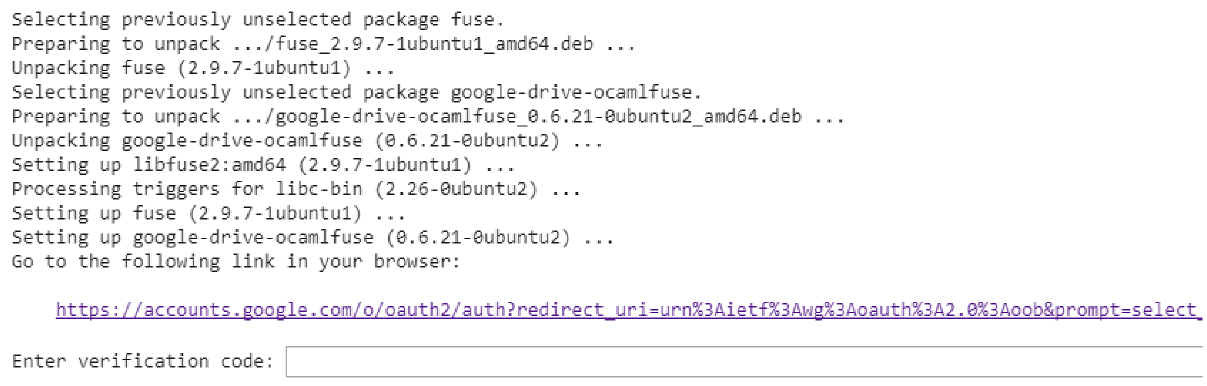


6. Working with Colab Notebook is very similar to the jupyter notebook.
7. Now, you need to do some authorization and mount google drive. Add a new cell and run following code

```
!apt-get install -y -qq software-properties-common python-software-properties
module-init-tools
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
```

```
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}
< /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -
secret={creds.client_secret}
```

When you run the code above, you should see a result like below twice.



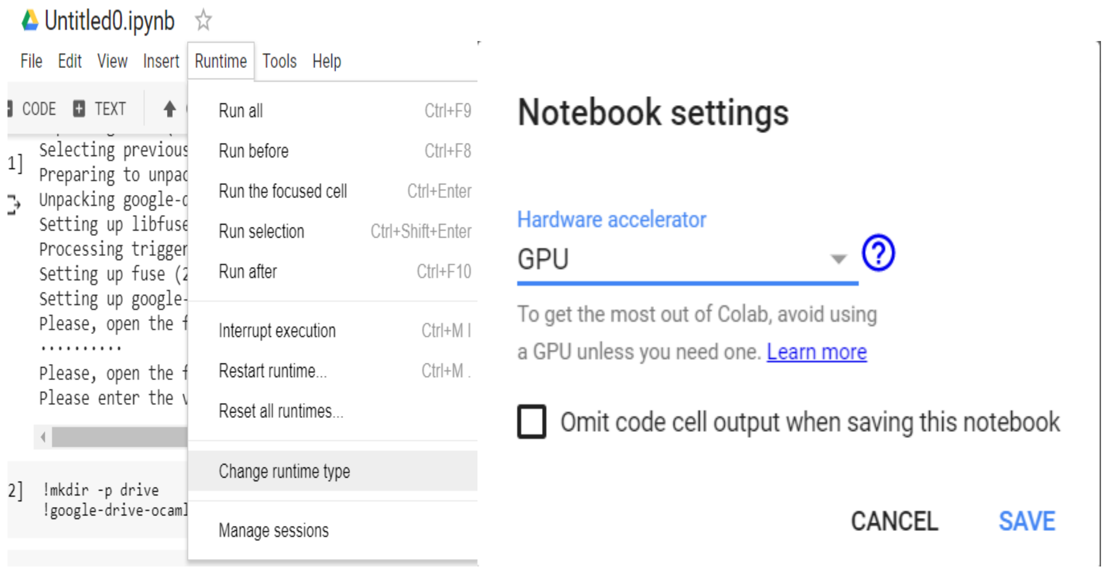
```
Selecting previously unselected package fuse.
Preparing to unpack .../fuse_2.9.7-1ubuntu1_amd64.deb ...
Unpacking fuse (2.9.7-1ubuntu1) ...
Selecting previously unselected package google-drive-ocamlfuse.
Preparing to unpack .../google-drive-ocamlfuse_0.6.21-0ubuntu2_amd64.deb ...
Unpacking google-drive-ocamlfuse (0.6.21-0ubuntu2) ...
Setting up libfuse2:amd64 (2.9.7-1ubuntu1) ...
Processing triggers for libc-bin (2.26-0ubuntu2) ...
Setting up fuse (2.9.7-1ubuntu1) ...
Setting up google-drive-ocamlfuse (0.6.21-0ubuntu2) ...
Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&prompt=select_

Enter verification code:
```

Click the link, copy verification code, paste it to text box and press enter. After completion of the authorization process, mount your Google Drive by running this code in a new code cell only once.

```
!mkdir -p drive
!google-drive-ocamlfuse drive
```

To check if it was successful, run `!ls drive` to view the list of items in your drive.

8. Now, change the runtime of the notebook by clicking "change runtime type" from the "Runtime" tab in the top menu bar, and click save after setting it as below.   .

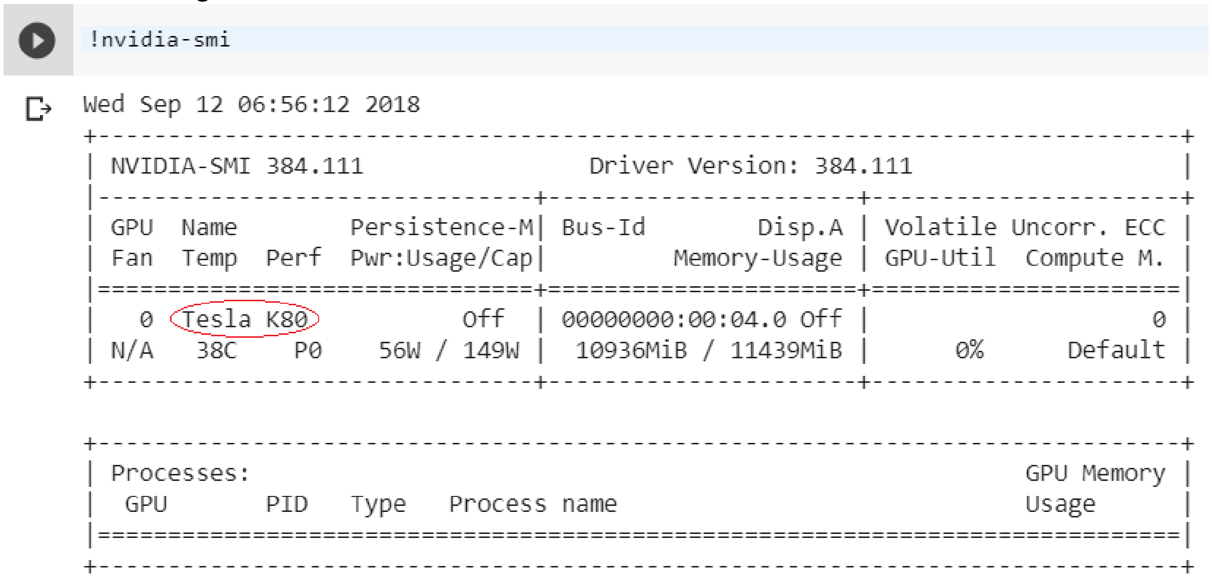


```
1]  Selecting previous
    Preparing to unpac
    Unpacking google-c
    Setting up libfuse
    Processing trigger
    Setting up fuse (2
    Setting up google-
    Please, open the f
    ..........
    Please, open the f
    Please enter the v

2]  !mkdir -p drive
    !google-drive-ocaml
```

Hardware accelerator
GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

9. Run !nvidia-smi to check GPU specifications by running !nvidia-smi. You will see something like this



```
!nvidia-smi
```

```
Wed Sep 12 06:56:12 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.111                Driver Version: 384.111                   |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   38C    P0    56W / 149W |  10936MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

2.2 Exercise

Let’s check if Keras is using GPU version of tensorflow by running this command in next code cell.

```
from keras import backend as K
K.tensorflow_backend.tf.config.list_physical_devices('GPU')
```

It should return output like this
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]

Now, you can follow exact same step as mentioned in section **1.2.** And compare execution time when you run the same code on CPU and GPU.