

# AI Deep Learning with Convolutional Neural Networks on Google Cloud Platform

**Thuan L. Nguyen**  
University of North Texas

*Big data can help established firms drastically transformed themselves, bring forth a whole new industry, and enable companies of any size to innovate, gain competitive advantage, and enhance business performance. However, how the valuable information hidden in vast volumes of data can be used to provide solutions to difficult practical problems such as computer vision, natural language processing problems, to name a few, is the matter. AI Deep learning, a subset of traditional machine learning, is based on the concepts of artificial neural networks and offers powerful methods to solve these problems.*

*Keywords: Artificial Intelligence (AI), Machine Learning, Deep Learning, Neural Networks, Artificial Neural Networks*

## INTRODUCTION

As a nascent field, big data or data science evolve with a warp speed, which has definitely caught the attention of scholars and practitioners in various industries (George & Lavie, 2016; McAfee & Brynjolfsson, 2012). The speed of evolving is intriguing enough to make academic researchers and business leaders wonder what emerging opportunities that big data and data science can offer (George & Lavie, 2016). Big data analytics has come out as a new important field of study for both researchers and practitioners, demonstrating the significant demand for solutions to business problems in a data-driven knowledge-based economy (Chen, Chiang, & Storey, 2012). However, how the valuable information hidden in vast volumes of data can be used to provide solutions to difficult practical problems such as computer vision, natural language processing problems, to name a few, is the matter.

Artificial intelligence (AI) fields such as machine learning and deep learning are rapidly developing fields and attempt to understand all the latest advancements of all these fields can be overwhelming. However, deep learning recently becomes much more popular thanks to its supremacy regarding accuracy in the attempts to train a huge amount of data, i.e., big data. There is a huge demand for deep learning skills in the research and the industry. Offering a course on deep learning not only makes an academic program of advanced data analytics much stronger regarding the curriculum but also provides students with critical knowledge and skills to be successful in the job market, now and the future.

### Deep Learning with TensorFlow

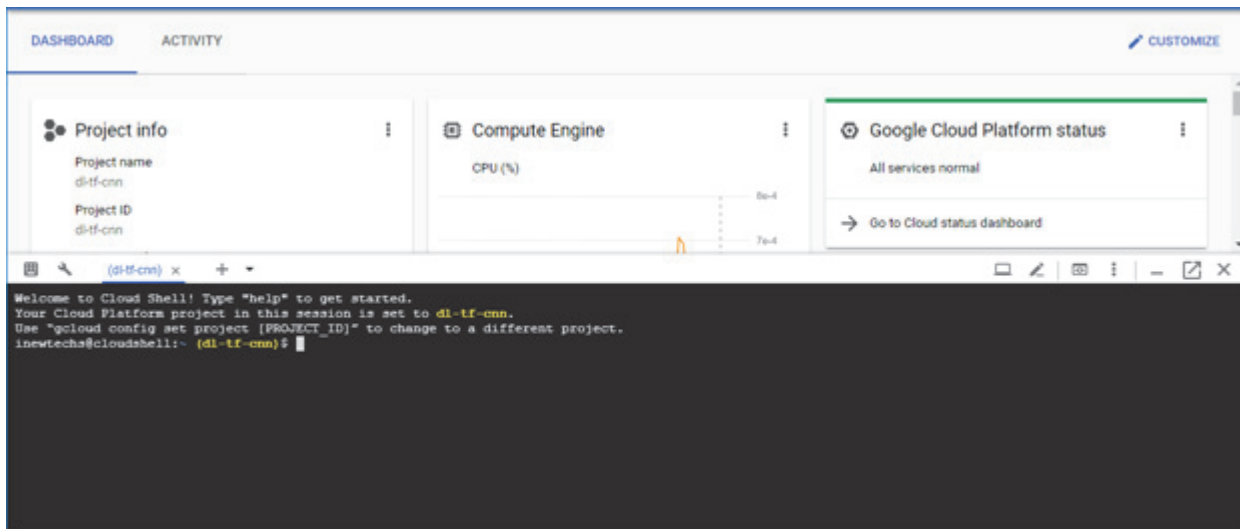
Deep learning is a machine learning mechanism characterized by a learning model consisting of several levels: The inputs of the most profound level are the output of the previous ones. At each level,

the inputs are transformed and become more and more abstract along with the process. This learning paradigm is modeled after how the human brain processes information and learns as responses to external stimuli.

TensorFlow is a powerful deep-learning open-source software library initially developed by Google Brain Team, a part of Google machine intelligence research organization, for Deep Neural Networks (DNNs) research (Gulli & Kapoor, 2017). The library is used for AI deep learning computation based on data flow graphs. In each graph, a node represents a mathematical operation while edges represent tensors that are multidimensional arrays of data (Gulli & Kapoor, 2017). For the last two years, TensorFlow has been the most popular deep learning framework for both academic research and industrial innovation (Hale, 2018).

## SET UP DEEP LEARNING SERVER on GOOGLE CLOUD PLATFORM

For individual users, using services offered on Google Cloud Platform (GCP) a Google account, or to be simpler, a Gmail account. The user needs to create one if he/she has not had it. The good thing is that any new user gets \$300.00 credit to be used on any service offered on the platform. Next, the user can set up a deep learning server set up the deep learning VM using the CLI (Command Line Interface) with the online CLOUD SHELL that can be accessed right under the GCP management console.



To set up a deep learning server using only CPU, not GPU, to save cost, the user can use the below script command line:

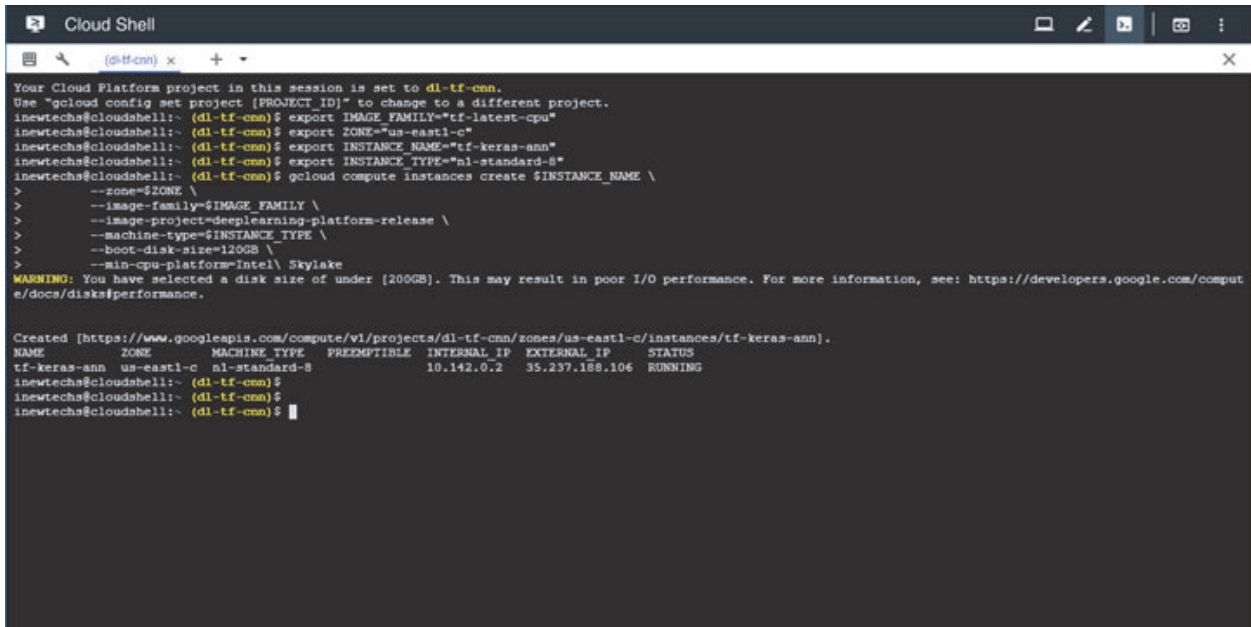
```
export IMAGE_FAMILY="tf-latest-cpu"  
export ZONE="us-east1-c"  
export INSTANCE_NAME="Name of the Instance"  
export INSTANCE_TYPE="n1-standard-8"  
gcloud compute instances create $INSTANCE_NAME \  
  --zone=$ZONE \  
  --image-family=$IMAGE_FAMILY \  
  --image-project=deeplearning-platform-release \  
  --machine-type=$INSTANCE_TYPE \  
  --boot-disk-size=120GB \  
  --min-cpu-platform=Intel\ Skylake
```

### IMPORTANT NOTES:

- > Replace “Name of the Instance” with the real name of the instance.
- > ANY NAME MUST BE: lower case, only letters (a – z), digits (0 -9), and hyphen (no underscore)
- > The user can name the instance whatsoever he/she wants.
- > The name MUST be embedded in the quotes as shown.
- > The user **MUST take note of this piece of information** that will be needed later.

### IMPORTANT NOTES:

- > The GCP region and zone: **us-east1-c** (Region: us-east1; Zone: c)
- > The user can select other regions and zones. However, this one is recommended.
- > The user **MUST take note of this piece of information** that will be needed later.



```
Cloud Shell
(dl-tf-cnn) x + - x
Your Cloud Platform project in this session is set to dl-tf-cnn.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
inewtechs@cloudshell:~ (dl-tf-cnn)$ export IMAGE_FAMILY="tf-latest-cpu"
inewtechs@cloudshell:~ (dl-tf-cnn)$ export ZONE="us-east1-c"
inewtechs@cloudshell:~ (dl-tf-cnn)$ export INSTANCE_NAME="tf-keras-ann"
inewtechs@cloudshell:~ (dl-tf-cnn)$ export INSTANCE_TYPE="n1-standard-8"
inewtechs@cloudshell:~ (dl-tf-cnn)$ gcloud compute instances create $INSTANCE_NAME \
> --zone=$ZONE \
> --image-family=$IMAGE_FAMILY \
> --image-project=deeplearning-platform-release \
> --machine-type=$INSTANCE_TYPE \
> --boot-disk-size=120GB \
> --min-cpu-platform=Intel\ Skylake
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https://developers.google.com/comput
e/docs/disks#performance.

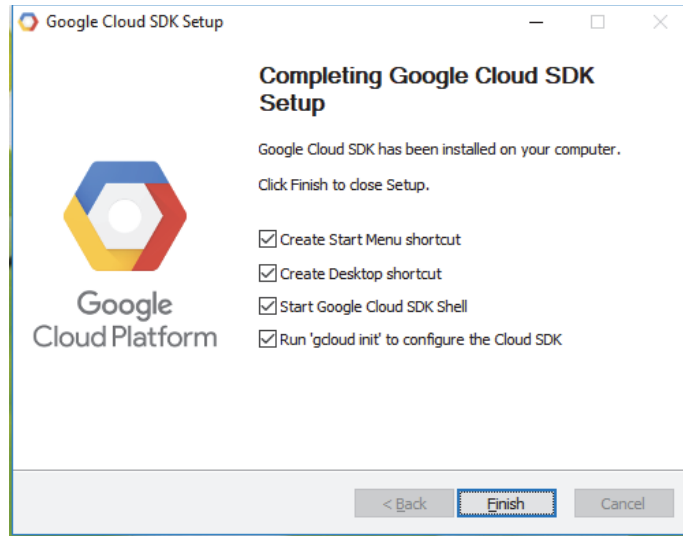
Created [https://www.googleapis.com/compute/v1/projects/dl-tf-cnn/zones/us-east1-c/instances/tf-keras-ann].
NAME        ZONE    MACHINE_TYPE  PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP  STATUS
tf-keras-ann  us-east1-c  n1-standard-8      10.142.0.2    35.237.158.106  RUNNING
inewtechs@cloudshell:~ (dl-tf-cnn)$
inewtechs@cloudshell:~ (dl-tf-cnn)$
inewtechs@cloudshell:~ (dl-tf-cnn)$
```

Running the script successfully will create a deep learning server with the above configuration and display its critical information in the Cloud Shell window as above.

### **INSTALL GCLOUD SDK and SET UP JUPYTER NOTEBOOK CONNECTIONS**

The Cloud SDK, or Gcloud, is a set of tools to access, work with, and manage services on GCP such as Google Compute Engine, Google App Engine, Google Cloud Storage, to name a few. To download and install Gcloud, the user can access this link: <https://cloud.google.com/sdk/docs/quickstart-windows>

When the user finishes installing the Gcloud SDK, he/she should run GCLOUD INIT as shown in the below dialogue to complete the installation process.



Jupyter Notebook is an open-source web application that can be used to create and share documents that contain live code (Python, R, and more), equations, data visualizations and text. The tool is considered as one of the most popular integrated development environments (IDE) for data cleaning, transformation and visualization, numerical computation and simulation, statistical modeling, machine learning and deep learning, and much more.

To start and use Jupyter Notebook to train deep learning models in GCP virtual machines, the user needs to do three steps:

**1. Step 1: Start Jupyter Notebook server in the GCP remote deep learning server**

- a. Start a GCloud SDK terminal window, say **Terminal #1**.
- b. Run the following SSH (Secure Shell) command line to connect to the remote server:

**`gcloud compute ssh USER@Instance_Name --project [PROJECT_ID] --zone [ZONE]`**

- c. In the remote server, create a new folder (Linux: `$ mkdir <folder name>`) under the home directory and change directory (Linux: `cd <folder name>`) to this newly created folder.
- d. Start Jupyter Notebook server by running the command line: **`$ jupyter notebook`**

```

inewtechs@tf-keras-ann: ~
individual files in /usr/share/doc/*/copyright.

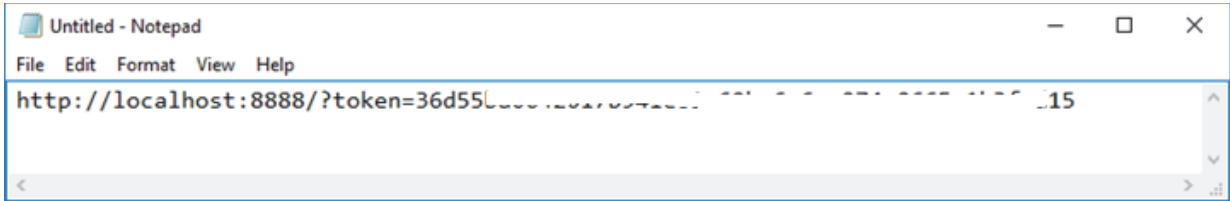
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
inewtechs@tf-keras-ann:~$
inewtechs@tf-keras-ann:~$
inewtechs@tf-keras-ann:~$ jupyter notebook
[I 17:28:49.577 NotebookApp] Writing notebook server cookie secret to /run/user/1001/jupyter/notebook_cookie_
secret
[I 17:28:51.731 NotebookApp] jupyter_tensorboard extension loaded.
[I 17:28:51.768 NotebookApp] JupyterLab extension loaded from /usr/local/lib/python3.5/dist-packages/jupyterl
ab
[I 17:28:51.768 NotebookApp] JupyterLab application directory is /usr/local/share/jupyter/lab
[I 17:28:51.770 NotebookApp] Serving notebooks from local directory: /home/inewtechs
[I 17:28:51.770 NotebookApp] The Jupyter Notebook is running at:
[I 17:28:51.770 NotebookApp] http://localhost:8888/?token=36d55ba0642017b941e81a69be6a6ae274e0665a1b3fed15
[I 17:28:51.770 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confi
rmation).
[W 17:28:51.771 NotebookApp] No web browser found: could not locate runnable browser.
[C 17:28:51.771 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=36d55ba0642017b941e81a69be6a6ae274e0665a1b3fed15

```

### **IMPORTANT NOTES:**

--> Copy this line by high-light it then paste it into a Notepad file (Windows) or any pure-text editor in any other OS platform.



--> **Pay attention to the details of this line:**

- This Jupyter Notebook server starts at the remote virtual machine (localhost: in the view of the remote VM).
- The server starts at the port 8888 by default.
- A token is issued for this Jupyter Notebook server.
- The token will be used for the authentication of connections to the remote Jupyter Notebook server.

### **IMPORTANT NOTES:**

--> LEAVE THIS SSH TERMINAL WINDOW AS IS: DON'T CLOSE IT

#### **2. Step 2: Forward a local port to the Port 8888 of the remote server in GCP**

- Start a GCLOUD SDK terminal window (**GCLOUD Terminal #2**)
- In the GCLOUD terminal window #2, SSH to the remote deep learning virtual machine
  - Run this command line:

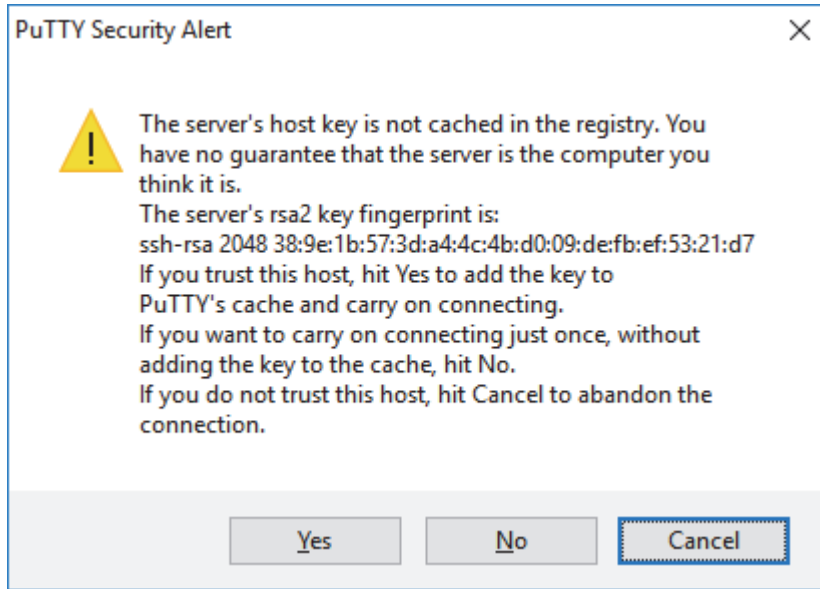
```
$ gcloud compute ssh USER@Instance_Name \  
    --project [PROJECT_ID] \  
    --zone [ZONE] \  
    -- -L 8000:localhost:8888
```

- USER: User name of the google account or the gmail account
  - E.g.: [john@gmail.com](mailto:john@gmail.com) → john: USER
- Instance\_Name: The name of the instance
  - E.g.: gcpVM1 → john@gcpVM1
- PROJECT\_ID: The project name
- ZONE: the region and zone
- 8000: port on the local computer
- 8888: port on the remote virtual machine

For example:

```
$ gcloud compute ssh john@gcpVM1 \  
    --project aProjName \  
    -- zone us-east1-c \  
    -- -L 8000:localhost:8888
```

```
Google Cloud SDK Shell
Welcome to the Google Cloud SDK! Run "gcloud -h" to get the list of available commands.
---
C:\APPLS\GOOGLE\GCP_SDK>
C:\APPLS\GOOGLE\GCP_SDK>
C:\APPLS\GOOGLE\GCP_SDK>gcloud compute ssh inewtechs@tf-keras-ann --project dl-tf-cnn --zone us-east1-c -- -L 8000:localhost:8888
```



- Click either YES or NO
  - An SSH terminal window pops up
  - **NOTES:** This is SSH TERMINAL WINDOW #2

```
inewtechs@tf-keras-ann: ~
Authenticating with public key "DESKTOP-MTT2KAM\Admin@DESKTOP-MTT2KAM"
=====
Welcome to the Google Deep Learning VM
=====

Version: ml3
Based on: Debian GNU/Linux 9.6 (stretch) (GNU/Linux 4.9.0-8-amd64 x86_64\n)

Resources:
* Google Deep Learning Platform StackOverflow: https://stackoverflow.com/questions/tagged/google-dl-platform
* Google Cloud Documentation: https://cloud.google.com/deep-learning-vm
* Google Group: https://groups.google.com/forum/#!forum/google-dl-platform

To reinstall Nvidia driver (if needed) run:
sudo /opt/deeplearning/install-driver.sh
TensorFlow comes pre-installed with this image. To install TensorFlow binaries in a virtualenv (or conda env), please use the binaries that are pre-built for this image. You can find the binaries at /opt/deeplearning/binaries/tensorflow/
If you need to install a different version of Tensorflow manually, use the command Deep Learning image with the right version of CUDA

Linux tf-keras-ann 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

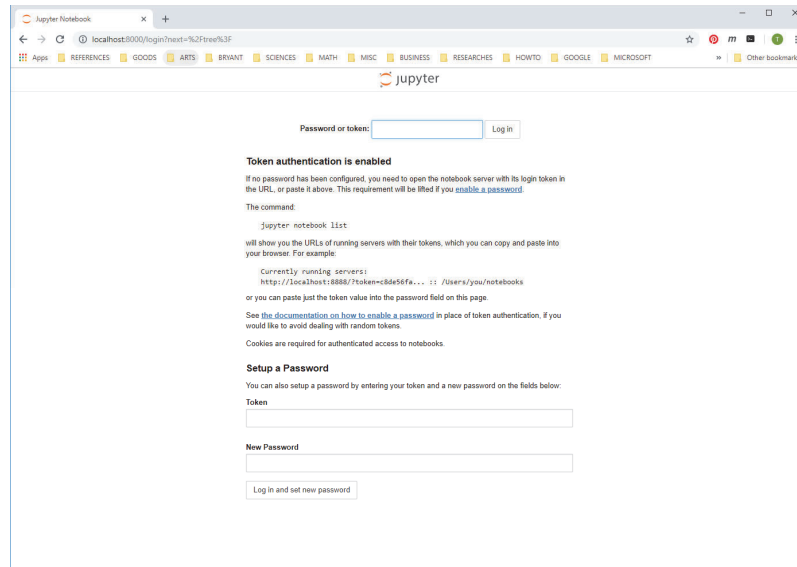
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.
inewtechs@tf-keras-ann:~$
```

## **IMPORTANT NOTES:**

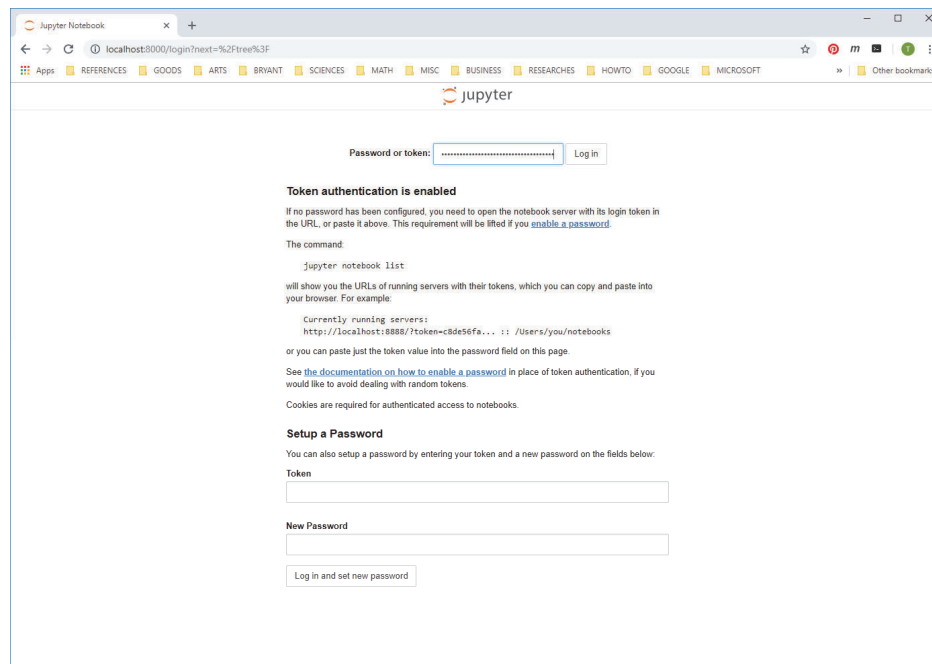
--> LEAVE THIS SSH TERMINAL WINDOW AS IS: DON'T CLOSE IT

3. Connect to the Jupyter Notebook server in the remote deep learning server in GCP

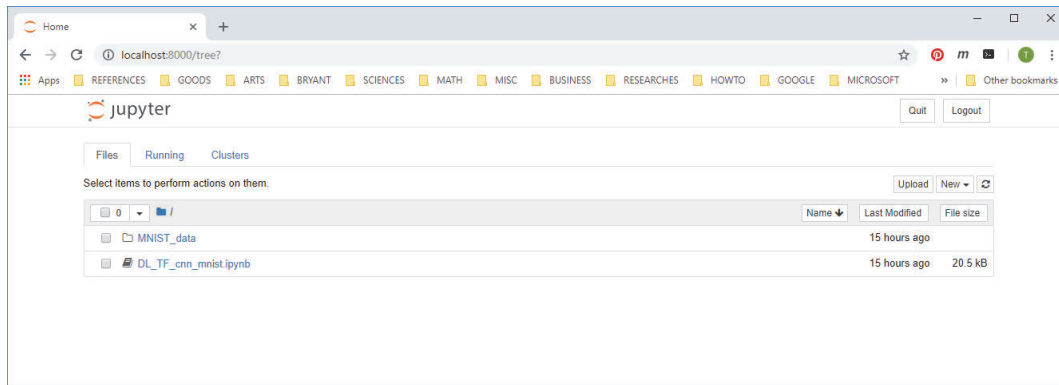
- Open Chrome browser
- Enter: <http://localhost:8000>
  - Into the URL search box



- Copy the token that has been copied into the aforementioned notepad file into the text field at the top



- Click Log in



## AN IMAGE RECOGNITION PROJECT WITH THE MNIST DATASET

MNIST (Mixed National Institute of Standards and Technology) database is dataset for handwritten digits. There are totally 70,000 images in the dataset in which 55000 images are the training data (mnist.train) – 55000, and the test data (mnist.test) contain 10000 images. The dataset consists of pairs – “handwritten digit image” and “label”:

- Digit ranges from 0 to 9.
- Handwritten digit image: Grayscale images with size 28 x 28 pixels.
- Label : Actual digit number THAT a digit image represents, either 0 to 9.

The project aims to build a convolutional neural network that has the following main layers in the order:

- The convolution layer 1 with ReLU (Rectified Linear Unit) as the activation function
- The pooling layer 1
- The convolution layer 2 with ReLU (Rectified Linear Unit) as the activation function
- The pooling layer 2
- And the fully-connected layers

## BUILD THE CONVOLUTIONAL NEURAL NETWORK

Import Python and TensorFlow libraries

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

import tensorflow as tf

from tensorflow.examples.tutorials.mnist import input_data
```

Load the MNIST dataset:

```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```



Define supporting functions:

```
def init_weights(shape):
    init_random_dist = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(init_random_dist)

def init_bias(shape):
    init_bias_vals = tf.constant(0.1, shape=shape)
    return tf.Variable(init_bias_vals)

def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
```

```
def convolutional_layer(input_x, shape):
    W = init_weights(shape)
    b = init_bias([shape[3]])
    return tf.nn.relu(conv2d(input_x, W) + b)

def max_pool_2by2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

```
def normal_full_layer(input_layer, size):
    input_size = int(input_layer.get_shape()[1])
    W = init_weights([input_size, size])
    b = init_bias([size])
    return tf.matmul(input_layer, W) + b
```

Build the convolutional neural network: Create placeholders and reshape the inputs

```
x = tf.placeholder(tf.float32, shape=[None, 784])
y_true = tf.placeholder(tf.float32, [None, 10])
x_image = tf.reshape(x, [-1, 28, 28, 1])
```

Build the convolutional neural network: Create convolution layers and pooling layers

```
convo_1 = convolutional_layer(x_image, shape=[5, 5, 1, 32])
convo_1_pooling = max_pool_2by2(convo_1)

convo_2 = convolutional_layer(convo_1_pooling, shape=[5, 5, 32, 64])
convo_2_pooling = max_pool_2by2(convo_2)
```

Build the convolutional neural network: Flatten the outputs of the last pooling layer

```
convo_2_flat = tf.reshape(convo_2_pooling, [-1, 7 * 7 * 64])
```

Build the convolutional neural network: Create fully-connected layers

```
full_layer_one = tf.nn.relu(normal_full_layer(convo_2_flat, 1024))

hold_prob = tf.placeholder(tf.float32)
full_one_dropout = tf.nn.dropout(full_layer_one, keep_prob=hold_prob)

y_pred = normal_full_layer(full_one_dropout, 10)
```

Build the convolutional neural network: Define the loss function and create optimizer and trainer

```
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels=y_true, logits=y_pred))

optimizer = tf.train.AdamOptimizer(learning_rate=0.001)

trainer = optimizer.minimize(cross_entropy)
```

## TRAIN AND TEST THE NEURAL NETWORK

```
init = tf.global_variables_initializer()

steps = 5000

with tf.Session() as sess:
    sess.run(init)

    for i in range(steps):

        # Each batch: 50 images
        batch_x, batch_y = mnist.train.next_batch(50)

        # Probability: 0.5 --> probability that the element is kept
        sess.run(trainer, feed_dict={x: batch_x, y_true: batch_y, hold_prob: 0.5})

        # Run this block of code for each 100 times of training, each time run a batch
        if i % 100 == 0:
            print('ON STEP: {}'.format(i))
            print('ACCURACY: ')

            # Compare to find matches of y_pred and y_true
            matches = tf.equal(tf.argmax(y_pred, 1), tf.argmax(y_true, 1))
            acc = tf.reduce_mean(tf.cast(matches, tf.float32))

            print(sess.run(acc, feed_dict={x: mnist.test.images, y_true: mnist.test.labels, hold_prob: 1.0}))
            print('\n')
```

## REFERENCES

- Chen, H., Chiang, R., & Storey, V. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, 36(4), 1165 – 1188.
- Ferkoun, M. (2014). *Cloud computing and big data: An ideal combination*. Retrieved from <https://www.ibm.com/blogs/cloud-computing/2014/02/cloud-computing-and-big-data-an-ideal-combination/>
- Gartner. (2015). *Gartner survey shows more than 75 percent of companies are investing or planning to invest in big data in the next two years*. Retrieved from <http://www.gartner.com/newsroom/id/3130817>.
- Gulli, A., & Kapoor, A. (2017). *TensorFlow 1.x Deep Learning Cookbook*. Birmingham, UK: Packt Publishing.
- Hale, J. (2018). *Deep Learning Framework Power Scores 2018*. Retrieved from <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>.
- McAfee, A., & Brynjolfsson, E. (2012). Big data: The management revolution. *Harvard Business Review*, 90, 61 – 67.
- Stourm, L., & Ebbes, P. (2017). *Analytics in the Era of Big Data: Opportunities and Challenges*. Retrieved from <http://www.hec.edu/Knowledge/Point-of-View/Analytics-in-the-Era-of-Big-Data-Opportunities-and-Challenges>
- Trovati, M., Hill, R., Anjum, A., Zhu, S. Y., & Liu, L. (2016). *Big-Data Analytics and Cloud Computing: Theory, Algorithms and Applications*. New York, NY: Springer.