STUDENTSPLAYGROUND

.

Phase 1:

Understanding Image Recognition using CIFAR 10

Technical Documentation

TABLE OF CONTENTS

0 PREFACE

0.1 PURPOSE OF THIS DOCUMENT

This document is a generic technical documentation for use by the project STUDENTSPLAYGROUND under the id : studentsone. It provides guidance for the members associated with this project as it shows the errors and mistakes while performing basic steps in image processing and also assists better ways to avoid wrong methods adopted while practising image processing and classification.

0.2 OVERVIEW

This documentation is basically a record of all the problems encountered during the initial learning phase of object detection which is image classification along with their solutions.

0.3 BASIS OF THIS DOCUMENT

This documentation is based on the practical implementation of basic image processing algorithm and image classification by the members of the group: Prasoon Kumar (leader), Hritik Kumar, Shivam Shubham under the esteemed guidance and direction of Dr. Sudhir Kumar Singh (Founder CEO of INVII.AI).

INTRODUCTION

PURPOSE OF THE PROJECT

- The project Studentsplayground, unlike to what the name suggests, actually suggests the play area for all its members who are actually intended to play with various algorithms and then come up with an efficient one for the main project.
- In the initial phase, the team worked on CIFAR-10 dataset and and training testing and evaluating the model through tensorflow checking different possibilities to find an even greater algorithm in terms of accuracy and an economical one with respect to time.

ACRONYMS AND ABBREVIATIONS

- This section should define all terms, acronyms and abbreviations used in this document:

c10	CIFAR -10 (basic data set the team worked on initially)
tflow	Tensorflow (a machine learning library)

gcp	Google Cloud Platform (a google based platform serving as a remote server provider for storage, computing and networking)
inst	Instance (a virtual machine under gcp projects)
bkt	Bucket(everything that's stored in google cloud are contained in a bucket)
ML	Machine learning

REFERENCES

- This section lists all the applicable and reference documents and learning materials :
- https://www.tensorflow.org/
- https://askubuntu.com/questions/1072683/how-can-i-install-protoc-on-ubuntu-16-04
- https://linuxize.com/
- https://www.youtube.com/watch?v=eYj3a9RsA5s HYPERLINK "https://www.youtube.com/watch?v=eYj3a9RsA5s&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44"& HYPERLINK "https://www.youtube.com/watch?v=eYj3a9RsA5s&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44"list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44
- https://www.youtube.com/watch?v=3BXfw_1_TF4
- https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
- https://stackoverflow.com/

TOOLS OVERVIEW

- This section should briefly introduce the tools and discuss the background to the project.

TENSORFLOW

- Tflow is an open source ML library for research and production by providing APIs to develop. The high-level Keras API provides building blocks to create and train deep learning models.
- https://www.tensorflow.org/tutorials/

IMAGE RECOGNITION

- Image recognition is used to perform a large no. of machine based visual tasks like labelling images with tags, guiding autonomous robots and what not. It is basically the ability to identify objects and motions.
- https://searchenterpriseai.techtarget.com/definition/image-recognition

GOOGLE CLOUD PLATFORM

- Gcp, is a suite of cloud computing services that runs on the same infrastructure that google uses internally for its end-user products such as google maps, YouTube etc.

Advantages of Google Cloud

- **Higher Productivity is gained through Quick Access to Innovation**: Google's systems can distribute updates efficiently and deliver functionality on a weekly basis or even faster.
- **Less Disruption is Caused When Users Adopt New Functionality**: Rather than large disruptive batches of change, Google delivers manageable improvements in a continuous stream.
- **Employees Can Work From Anywhere**: They can gain full access to information across devices from anywhere in the world through web-based apps powered by Google cloud.
- **Google Cloud Allows Quick Collaboration**: Many users can contribute to and access projects at the same time as data is stored in the cloud instead of their computers.
- **Customers are protected by Google's Investments in Security**: They are benefited by the process-based and physical security investments made by Google. Google hires the leading security experts in the world.
- **Less Data has to be stored on Vulnerable Devices**: Minimal data is stored on computers that may get compromised after a user stops using web-based apps on the cloud.
- **Customers get Higher Uptime and Reliability**: If a data center is not available for some reason, the system immediately falls back on the secondary center without any service interruption being visible to users.
- **Control and Flexibility Available to Users**: They have control over technology and data and have ownership over their data in Google apps. If they decide not to use the service any more, they can get their data out of Google cloud.
- **Google's Economies of Scale Let Customers Spend Less**: Google minimizes overheads and consolidates a small number of server configurations. It manages these through an efficient ratio of people to computers.
- https://console.cloud.google.com/getting-started?authuser=2 HYPERLINK "https://console.cloud.google.com/getting-started?authuser=2&project=studentsplayground"& HYPERLINK "https://console.cloud.google.com/getting-started?authuser=2&project=studentsplayground"project=studentsplayground

CIFAR 10

- The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

  The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

- https://www.tensorflow.org/tutorials/images/deep_cnn

PHASE (I) OF THE PROJECT

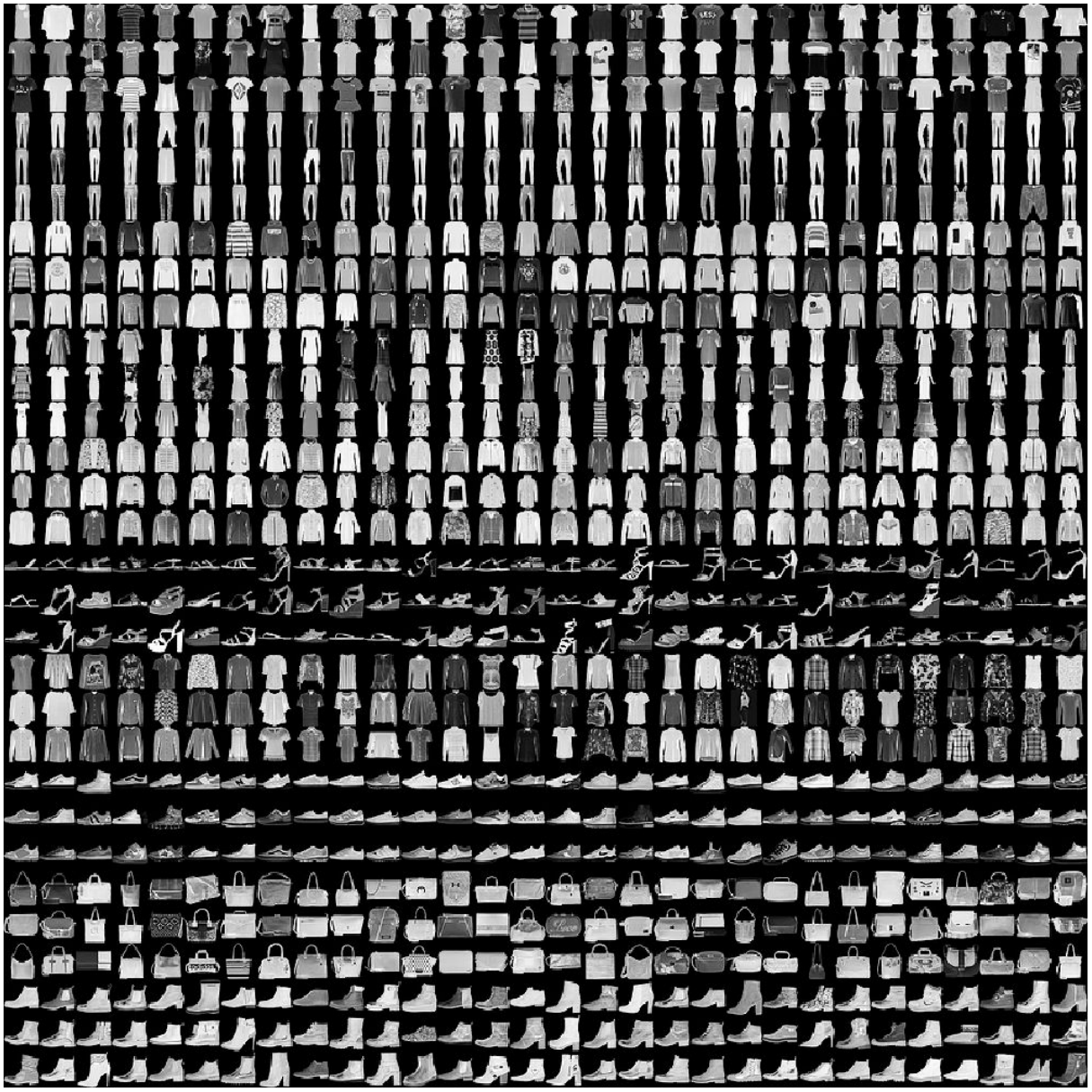
This section is intended to provide a detailed explanation of the steps involved, the errors encountered, the solutions to those error while applying the tflow model at every stage of the implementation.

ML USING TFLOW

Basic Classification

- Our 1st neural network was trained through tflow using tf.keras, a high level API for training tflow models.
- https://linuxize.com/post/how-to-install-tensorflow-on-ubuntu-18-04/ t

  This tutorial and guide was followed to install python3 on ubuntu and then install tflow in the terminal using pip.
- MNIST dataset was used to build, train and evaluate the accuracy of the tflow model.



- Ran the following code on the terminal to make sure that the tflow is working fine:

```
from __future__ import absolute_import, division, print_function,
unicode_literals
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
```

- it did run fine and gave the output:
- **1.14.0**
- So, we proceeded with further steps involved by following the tflow guide:

- https://www.tensorflow.org/tutorials/keras/basic_classification

Overfitting & Underfitting

- After evaluating the results and the accuracy of the model, it was concluded that the accuracy of our model on the validation data would peak after training for a number of epochs, and would then start decreasing.

  In other words, our model would overfit to the training data. Learning how to deal with overfitting is important. Although it's often possible to achieve high accuracy on the training set, what we really want is to develop models that generalize well to a testing set (or data they haven't seen before).

  The opposite of overfitting is underfitting. Underfitting occurs when there is still room for improvement on the test data. This can happen for a number of reasons: If the model is not powerful enough, is over-regularized, or has simply not been trained long enough. This means the network has not learned the relevant patterns in the training data. If you train for too long though, the model will start to overfit and learn patterns from the training data that don't generalize to the test data. We need to strike a balance. Understanding how to train for an appropriate number of epochs as we'll explore below is a useful skill. To prevent overfitting, the best solution is to use more training data. A model trained on more data will naturally generalize better. When that is no longer possible, the next best solution is to use techniques like regularization. These place constraints on the quantity and type of information your model can store. If a network can only afford to memorize a small number of patterns, the optimization process will force it to focus on the most prominent patterns, which have a better chance of generalizing well.

```
60000/60000 [==============================] - 4s 75us/sample - loss: 0.5018
- acc: 0.8241
Epoch 2/5
60000/60000 [==============================] - 4s 71us/sample - loss: 0.3763
- acc: 0.8643
Epoch 3/5
60000/60000 [==============================] - 4s 71us/sample - loss: 0.3382
- acc: 0.8777
Epoch 4/5
60000/60000 [==============================] - 4s 72us/sample - loss: 0.3138
- acc: 0.8846
Epoch 5/5
60000/60000 [==============================] - 4s 72us/sample - loss: 0.2967
- acc: 0.8897

<tensorflow.python.keras.callbacks.History at 0x7f65fb64b5c0>
```

-

ADVANCED CNN

- Advanced CNN involved understanding and implementing C10 classification problem where we intended to classify RGB 32 x 32 pixel images across 10 categories:
  `airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.`

Goals

The goal of this tutorial is to build a relatively small convolutional neural network(CNN) for recognizing images. In the process, this tutorial:

- Highlights a canonical organization for network architecture, training and evaluation.
- Provides a template for constructing larger and more sophisticated models.

The reason CIFAR-10 was selected was that it is complex enough to exercise much of TensorFlow's ability to scale to large models. At the same time, the model is small enough to train fast, which is ideal for trying out new ideas and experimenting with new techniques.

Highlights of the Tutorial

The CIFAR-10 tutorial demonstrates several important constructs for designing larger and more sophisticated models in TensorFlow:

- Core mathematical components
  including **tf.nn.conv2d** (wiki), **tf.nn.relu**(wiki), **tf.nn.max_pool** (wiki)
  and **tf.nn.local_response_normalization**(section 3.3 in AlexNet paper).
- Visualization of network activities during training, including input images, losses and distributions of activations and gradients.
- Routines for calculating the **tf.train.ExponentialMovingAverage** of learned parameters and using these averages during evaluation to boost predictive performance.
- Implementation of a **tf.train.exponential_decay** that systematically decrements over time.
- Prefetching input data to isolate the model from disk latency and expensive image pre-processing.
-

C10 TUTORIAL

The code for this tutorial resides in
https://github.com/tensorflow/models/tree/master/tutorials/image/cifar10/

the following sequence must be followed in order to successfully build the model, train the model and evaluate the model:

`cifar10_input.py`	Loads CIFAR-10 dataset using tensorflow-datasets library.

`cifar10.py`	Builds the CIFAR-10 model.
`cifar10_train.py`	Trains a CIFAR-10 model on a CPU or GPU.
`cifar10_multi_gpu_train.py`	Trains a CIFAR-10 model on multiple GPUs.
`cifar10_eval.py`	Evaluates the predictive performance of a CIFAR-10 model.

- We studied and tried to understand the codes as much as we could and then decided how to implement using gcp

WORKING ON GCP

In the account ‘ iinvi.ai ’ formed in gcp we created an inst and named it instance2 using this video:
https://www.youtube.com/watch?v=_Q0tRI5hMnc HYPERLINK "https://www.youtube.com/watch?v=_Q0tRI5hMnc&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44&index=3&t=22s"& HYPERLINK "https://www.youtube.com/watch?v=_Q0tRI5hMnc&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44&index=3&t=22s"list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44 HYPERLINK "https://www.youtube.com/watch?v=_Q0tRI5hMnc&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44&index=3&t=22s"& HYPERLINK "https://www.youtube.com/watch?v=_Q0tRI5hMnc&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44&index=3&t=22s"index=3 HYPERLINK "https://www.youtube.com/watch?v=_Q0tRI5hMnc&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44&index=3&t=22s"& HYPERLINK "https://www.youtube.com/watch?v=_Q0tRI5hMnc&list=PL9ooVrP1hQOFUm7TmkH1zk5xy75GAxV44&index=3&t=22s"t=22s
Installation of tensorflow-gpu in google cloud platform in ubuntu-18.04

-
- After creating a instance of ubuntu in GCP, we have to install nvidia-driver, cuda , cudNN and then tensorflow-gpu
-
- update ubuntu

  sudo apt-get update
  sudo apt-get upgrade
  sudo apt-get install gnome-shell

- install nvidia driver (https://www.nvidia.com/en-gb/data-center/gpu-accelerated-applications/tensorflow/)
-

  sudo add-apt-repository ppa:graphics-drivers/ppa
  sudo apt-get update
  sudo apt-get install nvidia-430
-

- install cuda-toolkit
- (https://developer.nvidia.com/cuda-downloads?target_os=Linux HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"& HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"target_arch=x86_64 HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"& HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"target_distro=Ubuntu HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"& HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"target_version=1804 HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"& HYPERLINK "https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork"target_type=debnetwork)

  Download cuda deb network file from above link

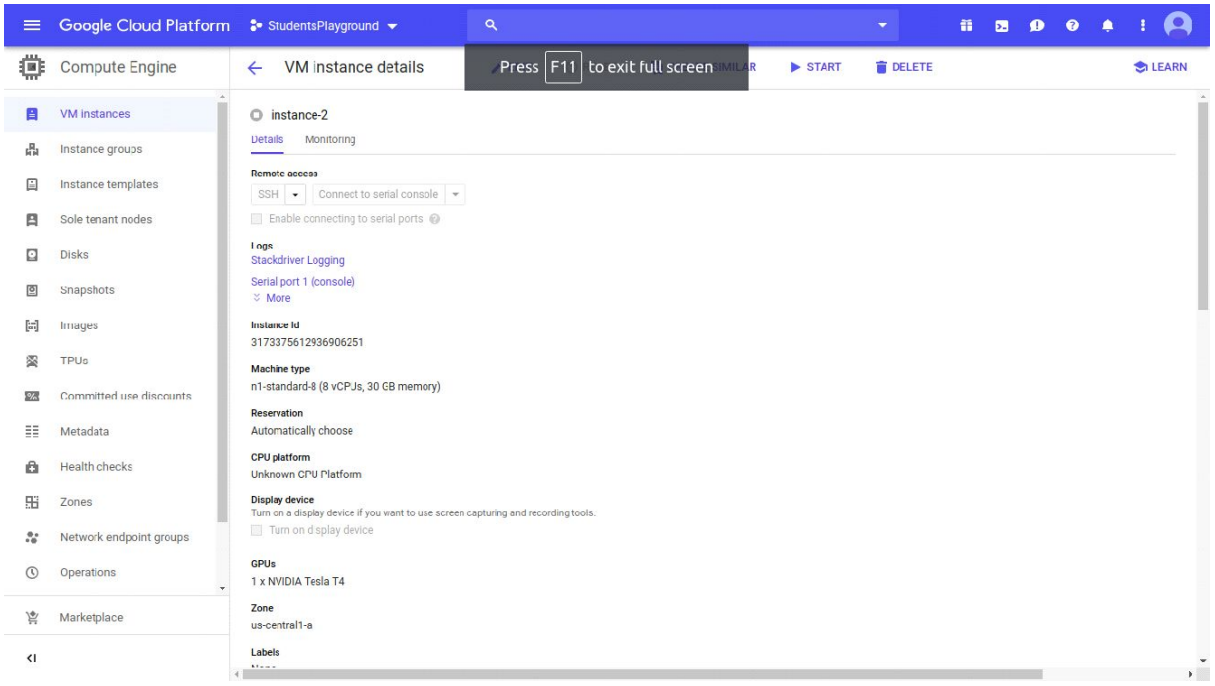
  sudo dpkg -i cuda-repo-ubuntu1804_10.0.130-1_amd64.deb(compatible so we downgrading to cuda 10.0)

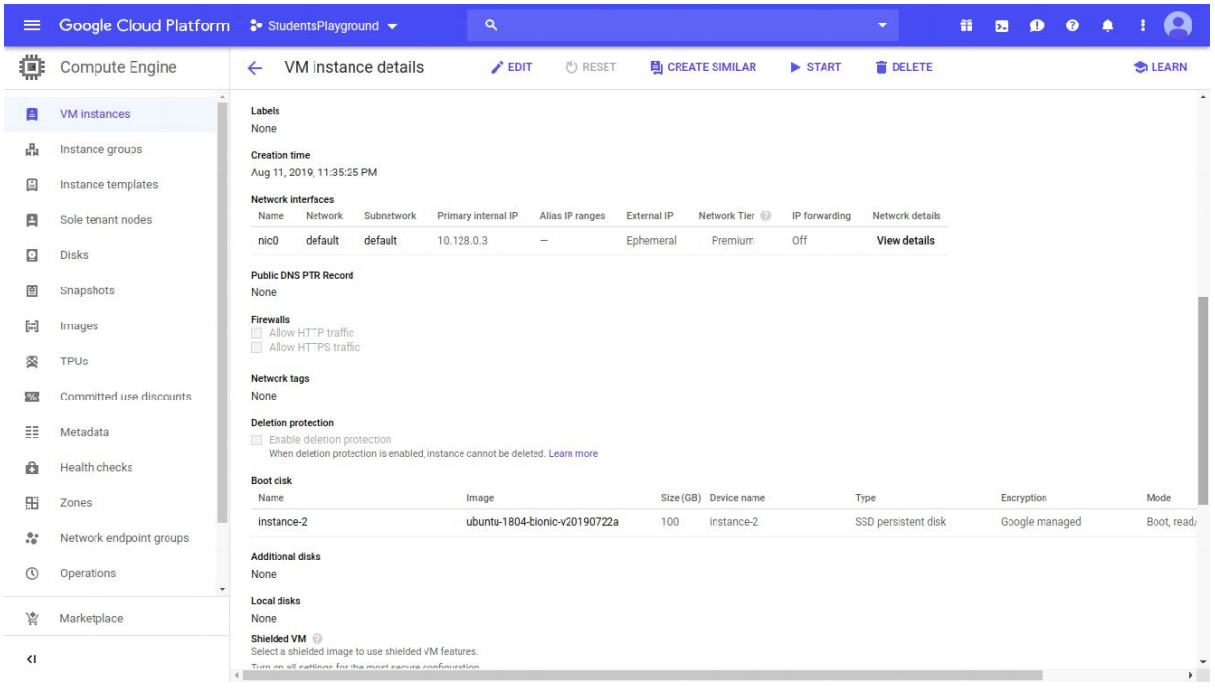
  sudo apt-key adv --fetch-keys (https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub)

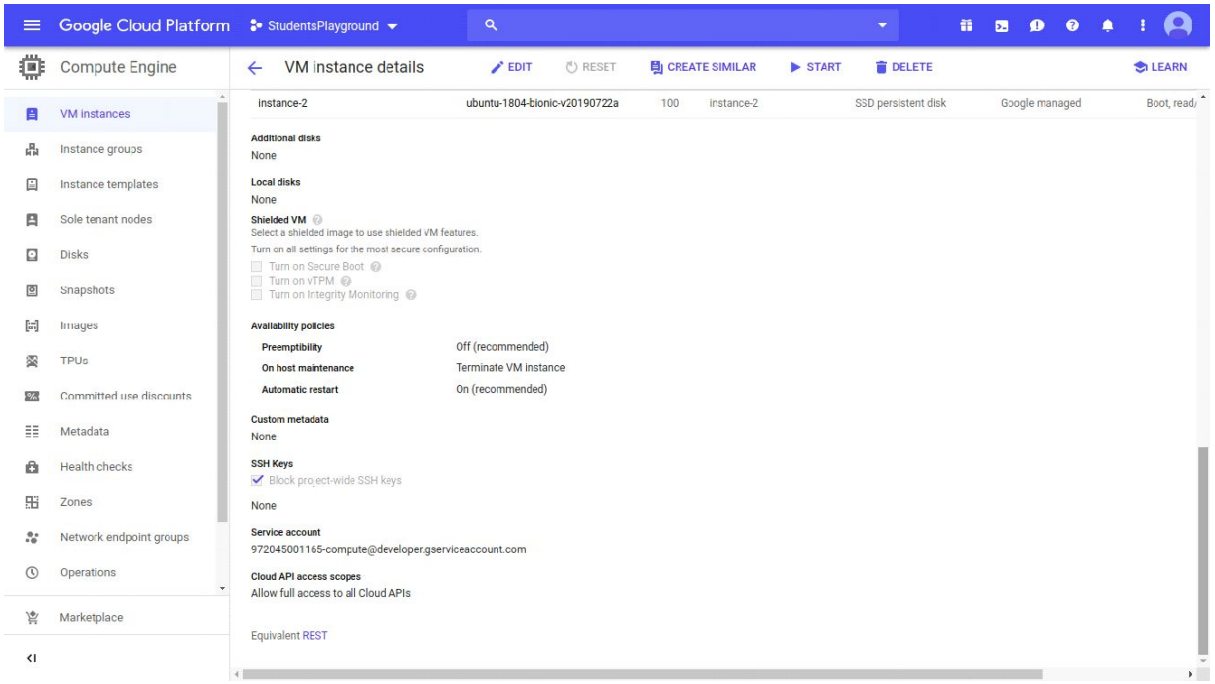
  sudo apt-get update

  sudo apt-get install cuda

- The details of our newly created inst is as shown in the screenshots below:


Google Cloud Platform
StudentsPlayground
Press F11 to exit full screen
Compute Engine
VM instance details
START
DELETE
LEARN
VM instances
Instance groups
Instance templates
Sole tenant nodes
Disks
Snapshots
Images
TPUs
Committed use discounts
Metadata
Health checks
Zones
Network endpoint groups
Operations
Marketplace
instance-2
Details
Monitoring
Remote access
SSH
Connect to serial console
Enable connecting to serial ports
Logs
Stackdriver Logging
Serial port 1 (console)
More
Instance Id
3173375612936906251
Machine type
n1-standard-8 (8 vCPUs, 30 GB memory)
Reservation
Automatically choose
CPU platform
Unknown CPU Platform
Display device
Turn on a display device if you want to use screen capturing and recording tools.
Turn on display device
GPUs
1 x NVIDIA Tesla T4
Zone
us-central1-a
Labels


Google Cloud Platform
StudentsPlayground
Compute Engine
VM instance details
EDIT
RESET
CREATE SIMILAR
START
DELETE
LEARN
VM instances
Instance groups
Instance templates
Sole tenant nodes
Disks
Snapshots
Images
TPUs
Committed use discounts
Metadata
Health checks
Zones
Network endpoint groups
Operations
Marketplace
Labels
None
Creation time
Aug 11, 2019, 11:35:25 PM
Network interfaces
Name Network Subnetwork Primary internal IP Alias IP ranges External IP Network Tier IP forwarding Network details
nic0 default default 10.128.0.3 — Ephemeral Premium Off View details
Public DNS PTR Record
None
Firewalls
Allow HTTP traffic
Allow HTTPS traffic
Network tags
None
Deletion protection
Enable deletion protection
When deletion protection is enabled, instance cannot be deleted. Learn more
Boot disk
Name Image Size (GB) Device name Type Encryption Mode
instance-2 ubuntu-1804-bionic-v20190722a 100 instance-2 SSD persistent disk Google managed Boot, read/
Additional disks
None
Local disks
None
Shielded VM
Select a shielded image to use shielded VM features.



after creating the inst, we installed python3, venv, pip and tflow in the compute engine created in the virtual machine that is instance2 like we did in the ubuntu terminal.

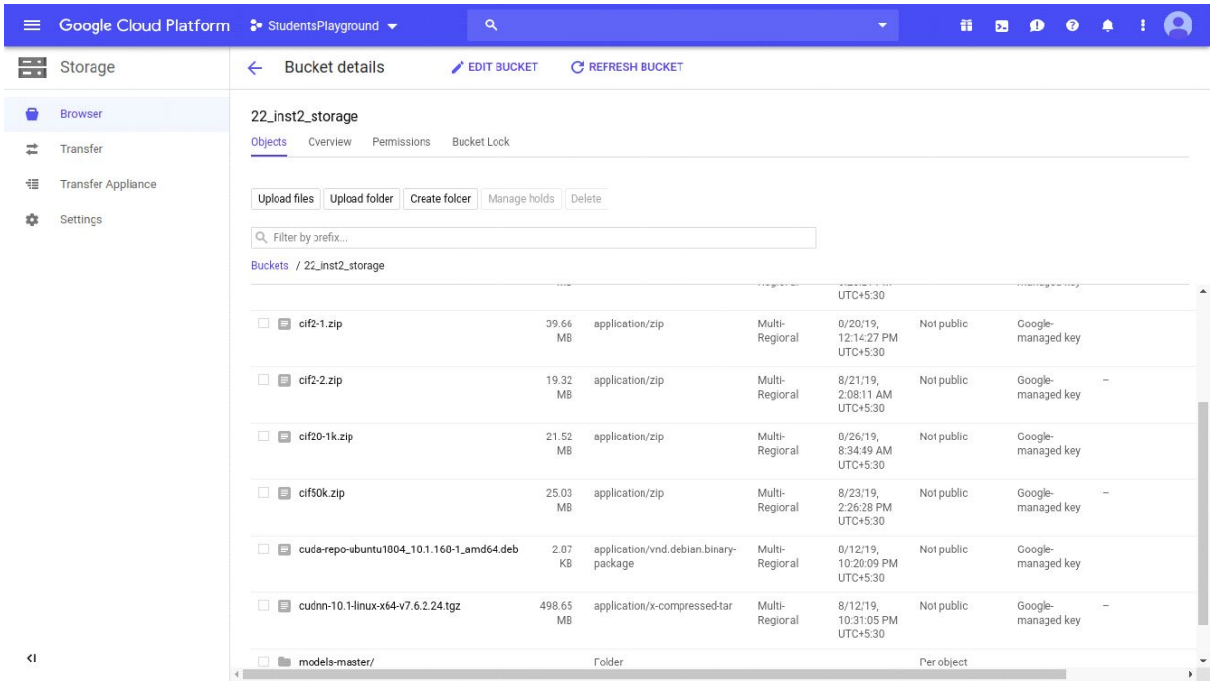
Before downloading cudNN.tgz file to your local PC, create a bucket in GCP so that we can directly import large file to our instance-shell

Create a bucket – multi regional – set object level and bucket level policy transfer bucket file to

instance.

we created a tflow directory in the inst and

to load the data into the directory for training the model we then created a bkt:



on connecting the bkt to the inst we uploaded those datasets into the inst from that bkt and then started working on the C10 datasets and tflow codes:

gsutil ls (to check your bucket is accessible or not, if not create a new one)

gcloud init (pick project_name – pick zone)

- install cudNN (for downloading cudNN, create a new nvidia account https://developer.nvidia.com/cudnn)

Copying cudnn file from bucket to instance
gsutil cp [file_source_URL] [file_destination_URL]

storage error (400 Bucket is requester pays bucket but no user project provided.)
gsutil -u [PROJECT_ID] cp gs://[bucket_name]/[obj_name] [obj_destination]

- unzip cudNN

tar -xzvf [cudnn filename .tgz]

- Copy the following files into the CUDA Toolkit directory, and change the file permissions

sudo cp cuda/include/cudnn.h /usr/local/cuda/include

sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*

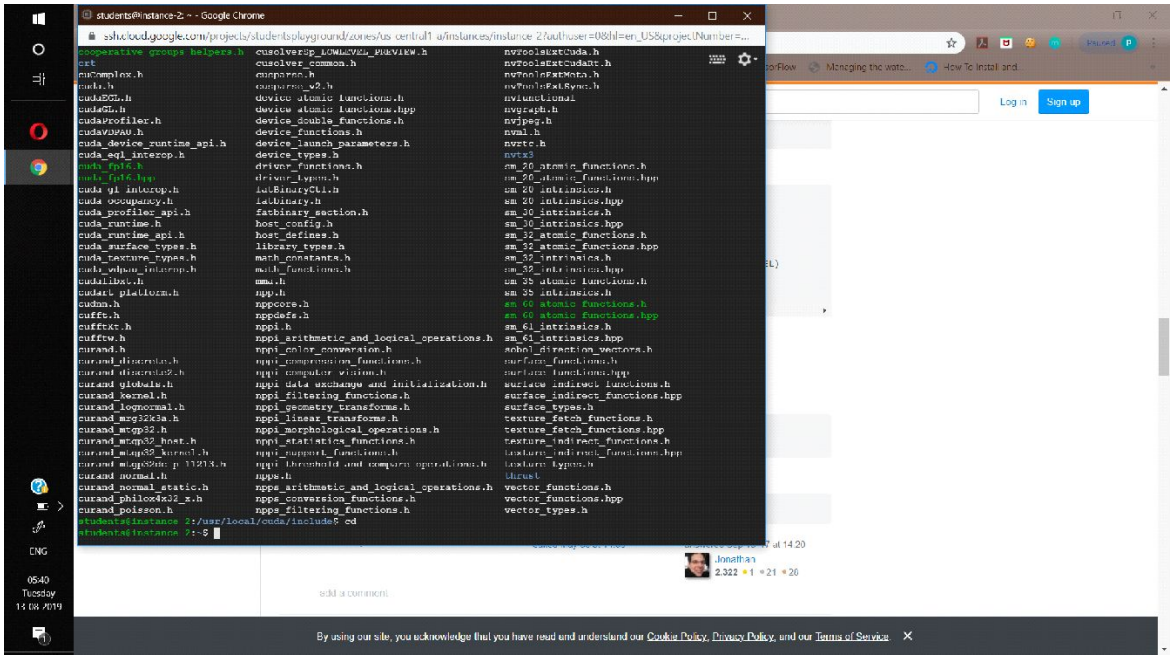
- Prepare for TensorFlow dependencies and required packages.
  sudo apt-get install libcupti-dev
- install tensorflow-gpu
  first install pip for python3(**sudo apt install python3-pip**)
  python3 -m pip install tensorflow-gpu

- open python3 and Verify tensorflow-gpu

  import tensorflow as tf
  print(tf.__version__)
  hello = tf.constant('Hello, TensorFlow!')
  sess = tf.Session()
  print(sess.run(hello)) O/P – 1.14 Hello, TensorFlow!'

- Install python libraries(numpy,pandas,matplotlib,scikit-learn,scipy,seaborn)
  python3 -m pip install [lib – name]
- check tensorflow-gpu
  tf.test.is_gpu_available(
  cuda_only=False,
  min_cuda_compute_capability=None) O/P – TRUE

These are the library files installed and needed:



1. To load cif10 dataset from tflow-datasets library, we ran the code in the file **cifar10_input.py** from models folder.
(took 6 seconds to run)
2. Running cifar10.py built the predefined model which will work on the loaded tfow dataset.

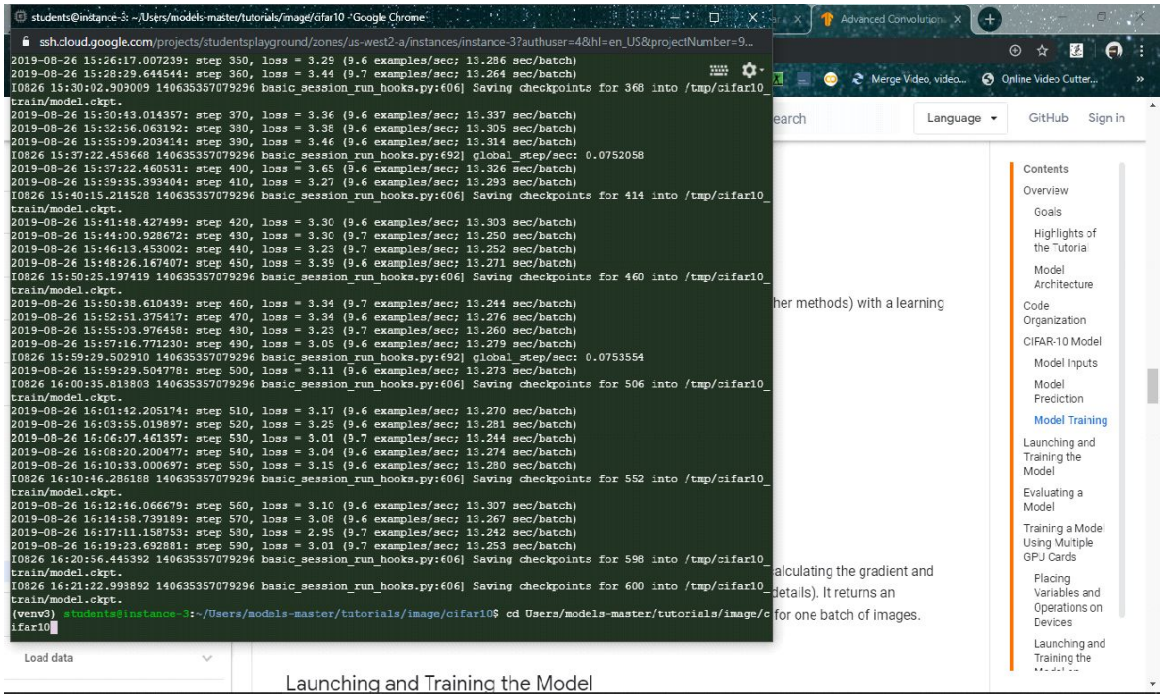
3. Next we ran cifar10_train.py to train the built model on the dataset in the GPU provided in GCP.
4. At last to evaluate the predictive performance of the tflow model we ran cifar10_eval.py.

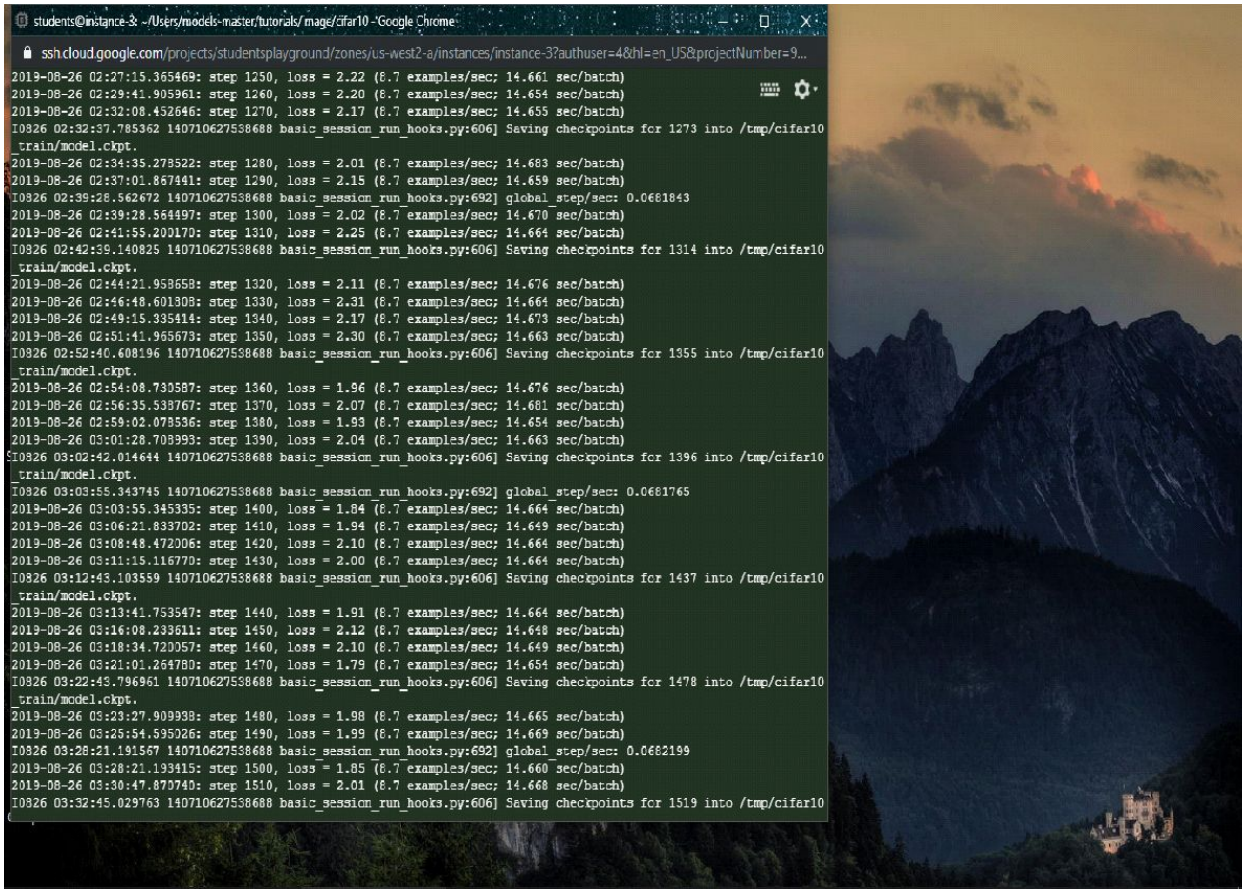
The convergence of the loss value is observed while training the model.
We get the accuracy after evaluating.
The value of loss can be considered for evaluating the accuracy of the model only when the value of loss converges after some no. of steps.

As the no. of steps for which the model has been trained increases the chances of the loss value converging increases :

No. of Steps	**Loss (Training)**	**Accuracy (evaluation)**
* 500 steps	arbitrary loss=3.6	accuracy = 56%
* 2000 steps	arbitrary loss=2.2	accuracy =68%
* 5000 steps	arbitrary loss=1.6	accuracy =72%
*** 10000 steps**	partly converged loss=1.1	accuracy =74%
* 20000 steps	partly converged loss=1.0	accuracy =79%
* 50000 steps	converged loss=0.7	accuracy = 83%

when the training part of the model runs, it appears like the image below:




students@instance-3: ~/Users/models-master/tutorials/image/cifar10 - Google Chrome
ssh.cloud.google.com/projects/studentsplayground/zones/us-west2-a/instances/instance-3?authuser=4&hl=en_US&projectNumber=9...
2019-08-26 02:27:15.365469: step 1250, loss = 2.22 (8.7 examples/sec; 14.661 sec/batch)
2019-08-26 02:29:41.905961: step 1260, loss = 2.20 (8.7 examples/sec; 14.654 sec/batch)
2019-08-26 02:32:08.452646: step 1270, loss = 2.17 (8.7 examples/sec; 14.655 sec/batch)
I0326 02:32:37.785362 140710627538688 basic_session_run_hooks.py:606] Saving checkpoints for 1273 into /tmp/cifar10_train/model.ckpt.
2019-08-26 02:34:35.278522: step 1280, loss = 2.01 (8.7 examples/sec; 14.683 sec/batch)
2019-08-26 02:37:01.867441: step 1290, loss = 2.15 (8.7 examples/sec; 14.659 sec/batch)
I0326 02:39:28.562672 140710627538688 basic_session_run_hooks.py:692] global_step/sec: 0.0681843
2019-08-26 02:39:28.564497: step 1300, loss = 2.02 (8.7 examples/sec; 14.670 sec/batch)
2019-08-26 02:41:55.200170: step 1310, loss = 2.25 (8.7 examples/sec; 14.664 sec/batch)
I0326 02:42:39.140825 140710627538688 basic_session_run_hooks.py:606] Saving checkpoints for 1314 into /tmp/cifar10_train/model.ckpt.
2019-08-26 02:44:21.958658: step 1320, loss = 2.11 (8.7 examples/sec; 14.676 sec/batch)
2019-08-26 02:46:48.601308: step 1330, loss = 2.31 (8.7 examples/sec; 14.664 sec/batch)
2019-08-26 02:49:15.335414: step 1340, loss = 2.17 (8.7 examples/sec; 14.673 sec/batch)
2019-08-26 02:51:41.965673: step 1350, loss = 2.30 (8.7 examples/sec; 14.663 sec/batch)
I0326 02:52:40.608196 140710627538688 basic_session_run_hooks.py:606] Saving checkpoints for 1355 into /tmp/cifar10_train/model.ckpt.
2019-08-26 02:54:08.730587: step 1360, loss = 1.96 (8.7 examples/sec; 14.676 sec/batch)
2019-08-26 02:56:35.538767: step 1370, loss = 2.07 (8.7 examples/sec; 14.681 sec/batch)
2019-08-26 02:59:02.078536: step 1380, loss = 1.93 (8.7 examples/sec; 14.654 sec/batch)
2019-08-26 03:01:28.708993: step 1390, loss = 2.04 (8.7 examples/sec; 14.663 sec/batch)
I0326 03:02:42.014644 140710627538688 basic_session_run_hooks.py:606] Saving checkpoints for 1396 into /tmp/cifar10_train/model.ckpt.
I0326 03:03:55.343745 140710627538688 basic_session_run_hooks.py:692] global_step/sec: 0.0681765
2019-08-26 03:03:55.345335: step 1400, loss = 1.84 (8.7 examples/sec; 14.664 sec/batch)
2019-08-26 03:06:21.833702: step 1410, loss = 1.94 (8.7 examples/sec; 14.649 sec/batch)
2019-08-26 03:08:48.472006: step 1420, loss = 2.10 (8.7 examples/sec; 14.664 sec/batch)
2019-08-26 03:11:15.116770: step 1430, loss = 2.00 (8.7 examples/sec; 14.664 sec/batch)
I0326 03:12:43.103559 140710627538688 basic_session_run_hooks.py:606] Saving checkpoints for 1437 into /tmp/cifar10_train/model.ckpt.
2019-08-26 03:13:41.753547: step 1440, loss = 1.91 (8.7 examples/sec; 14.664 sec/batch)
2019-08-26 03:16:08.233611: step 1450, loss = 2.12 (8.7 examples/sec; 14.648 sec/batch)
2019-08-26 03:18:34.720057: step 1460, loss = 2.10 (8.7 examples/sec; 14.649 sec/batch)
2019-08-26 03:21:01.264780: step 1470, loss = 1.79 (8.7 examples/sec; 14.654 sec/batch)
I0326 03:22:43.796961 140710627538688 basic_session_run_hooks.py:606] Saving checkpoints for 1478 into /tmp/cifar10_train/model.ckpt.
2019-08-26 03:23:27.909938: step 1480, loss = 1.98 (8.7 examples/sec; 14.665 sec/batch)
2019-08-26 03:25:54.595026: step 1490, loss = 1.99 (8.7 examples/sec; 14.669 sec/batch)
I0326 03:28:21.191567 140710627538688 basic_session_run_hooks.py:692] global_step/sec: 0.0682199
2019-08-26 03:28:21.193415: step 1500, loss = 1.85 (8.7 examples/sec; 14.660 sec/batch)
2019-08-26 03:30:47.870740: step 1510, loss = 2.01 (8.7 examples/sec; 14.668 sec/batch)
I0326 03:32:45.029763 140710627538688 basic_session_run_hooks.py:606] Saving checkpoints for 1519 into /tmp/cifar10