1. What is TensorFlow? Which company is the leading contributor to TensorFlow?

TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning faster and easier

TensorFlow is a Python-friendly open source library for numerical computation that makes machine learning faster and easier

Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.

As mentioned above, Google is the main contributor to TensorFlow

Source <https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html>

1. What is TensorRT? How is it different from TensorFlow?

Source <https://developer.nvidia.com/tensorrt>

NVIDIA TensorRT™ is an SDK for high-performance deep learning inference. It includes a deep learning inference optimizer and runtime that delivers low latency and high-throughput for deep learning inference applications.

TensorRT-based applications perform up to 40x faster than CPU-only platforms during inference. With TensorRT, you can optimize neural network models trained in all major frameworks, calibrate for lower precision with high accuracy, and finally deploy to hyperscale data centers, embedded, or automotive product platforms.

TensorRT is built on CUDA, NVIDIA’s parallel programming model, and enables you to optimize inference for all deep learning frameworks leveraging libraries, development tools and technologies in CUDA-X for artificial intelligence, autonomous machines, high-performance computing, and graphics.

Additional information by <https://github.com/ardianumam/Tensorflow/TensorRT/blob/master/1_convert_TF_to_TRT.ipynb>

TensorRT is an optimization tool provided by NVIDIA that applies graph optimization and layer fusion, and finds the fastest implementation of a deep learning model. In other words, TensorRT will optimize our deep learning model so that we expect a faster inference time than the original model (before optimization), such as 5x faster or 2x faster. The bigger model we have, the bigger space for TensorRT to optimize the model. Furthermore, this TensorRT supports all NVIDIA GPU devices, such as 1080Ti, Titan XP for Desktop, and Jetson TX1, TX2 for embedded device.

The difference between TensorRT and TensorFlow is that while the latter is used to write various ML and deep learning models, the former (TensorRT) allows one to optimize those models written in TensorFlow.

1. What is ImageNet? How many images does it contain? How many classes?

ImageNet is a project that organizes a large hand-annotated visual database designed of more than 14 million items, for visual object recognition and for use by deep learning models like Convolutional Neural Networks.

There are 21842 subcategories (classes) in ImageNet. These subcategories can be considered as sub-trees of 27 high-level categories. Thus, ImageNet is a well-organized hierarchy that makes it useful for supervised machine learning tasks.

Source Wikipedia and <https://devopedia.org/imagenet>

1. Please research and explain the differences between MobileNet and GoogleNet (Inception) architectures.

GoogleNet (a.k.a. Inception V1) was the winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 competition from Google. It achieved a top-5 error rate of 6.67%! This was very close to human level performance which the organizers of the challenge were now forced to evaluate. The network used a CNN inspired by LeNet but implemented a novel element which is dubbed an inception module. It used batch normalization, image distortions and RMSprop.

MobilleNet: Is a family of general purpose computer vision neural networks designed with mobile devices in mind to support classification, detection and more. The ability to run deep networks on personal mobile devices improves user experience, offering anytime, anywhere access, with additional benefits for security, privacy, and energy consumption. As new applications emerge allowing users to interact with the real world in real time, so does the need for ever more efficient neural networks.

Source <https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html>

1. In your own words, what is a bottleneck?

It is a layer with less neurons than the preceding layer, to encourage feature compression and feature reduction.

Answers for the below questions can referenced by the model performance graph in next page:

1. In the TF1 lab, you trained the last layer (all the previous layers retain their already-trained state). Explain how the lab used the previous layers (where did they come from? how were they used in the process?)
2. How does a low --learning\_rate (step 7 of TF1) value (like 0.005) affect the precision? How much longer does training take?
3. How about a --learning\_rate (step 7 of TF1) of 1.0? Is the precision still good enough to produce a usable graph?
4. For step 8, you can use any images you like. Pictures of food, people, or animals work well. You can even use [ImageNet](http://www.image-net.org/) images. How accurate was your model? Were you able to train it using a few images, or did you need a lot?
5. Run the TF1 script on the CPU (see instructions above) How does the training time compare to the default network training (section 4)? Why?
6. Try the training again, but this time do export ARCHITECTURE="inception\_v3" Are CPU and GPU training times different?

CPU Training Time was at least 300% more than GPU Training Time (GPU around 3 minutes vs. 10+ minutes for CPU)

1. Given the hints under the notes section, if we trained Inception\_v3, what do we need to pass to replace ??? below to the label\_image script? Can we also glean the answer from examining TensorBoard?

