1)what does a savemodel contain ?how do you inspect its content ?

Ans :SavedModel is a directory containing serialized signatures and the state needed to run them, including variable values and vocabularies. ls {mobilenet\_save\_path} assets saved\_model.pb variables. The saved\_model.

SavedModel contains a complete TensorFlow program, including trained parameters (i.e, tf.Variables) and computation. It does not require the original model building code to run, which makes it useful for sharing or deploying with TFLite, TensorFlow.js, TensorFlow Serving, or TensorFlow Hub.

Save: tf.saved\_model.save(model, path\_to\_dir)

Load: model = tf.saved\_model.load(path\_to\_dir).

2)when should you use TF serving ? What are its main Features ?what are some tools you can use to deploy it ?

Ans : TensorFlow Serving is a flexible, high-performance serving system for machine learning models, designed for production environments. TensorFlow Serving makes it easy to deploy new algorithms and experiments, while keeping the same server architecture and APIs.

Deploy tools :

Step 1: Install the Docker App.

Step 2: Pull the TensorFlow Serving Image. docker pull tensorflow/serving. ...

Step 3: Create and Train the Model. ...

Step 4: Save the Model. ...

Step 5: Serving the model using Tensorflow Serving. ...

Step 6: Make a REST request the model to predict.

3)how do you deploy a model across multiple TF serving instances ?

Ans : Step 1: Install the Docker App.

Step 2: Pull the TensorFlow Serving Image. docker pull tensorflow/serving. ...

Step 3: Create and Train the Model. ...

Step 4: Save the Model. ...

Step 5: Serving the model using Tensorflow Serving. ...

Step 6: Make a REST request the model to predict.

4)when should you use the grpc API rather than the REST API to query a model served by TF serving ?

Ans : RPC (Remote Procedure Call) is classic and the oldest API style currently in use. It uses procedure calls to request a service from a remote server the same way you would request a local system — via direct actions to the server (like SendUserMessages, LocateVehicle, addEntry). RPC is an efficient way to build APIs; RPC messages are lightweight and the interactions are straightforward.

gRPC benefits :

gRPC offers a refreshed take on the old RPC design method by making it interoperable, modern, and efficient using such technologies as Protocol Buffers and HTTP/2. The following benefits make it a solid candidate for replacing REST in some operations.gRPC owes its success to the employment of two techniques: using HTTP/2 instead of HTTP/1.1 and protocol buffers as an alternative to XML and JSON. Most of the gRPC benefits stem from using these technologies.

gRPC is roughly 7 times faster than REST when receiving data & roughly 10 times faster than REST when sending data for this specific payload. This is mainly due to the tight packing of the Protocol Buffers and the use of HTTP/2 by gRPC.so we used grpc model rather than RESTAPI.

5)what are the different ways TFLite reduces a models size to make it run on mobile or embedded device ?

Ans : TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and edge devices.

Optimized for on-device machine learning, by addressing 5 key constraints: latency (there's no round-trip to a server), privacy (no personal data leaves the device), connectivity (internet connectivity is not required), size (reduced model and binary size) and power consumption (efficient inference and a lack of network connections).

Multiple platform support, covering Android and iOS devices, embedded Linux, and microcontrollers.

Diverse language support, which includes Java, Swift, Objective-C, C++, and Python.

High performance, with hardware acceleration and model optimization.

End-to-end examples, for common machine learning tasks such as image classification, object detection, pose estimation, question answering, text classification, etc. on multiple platforms.

6) what is quantization aware training and why would you need it ?

Ans :

Quantization aware training emulates inference-time quantization, creating a model that downstream tools will use to produce actually quantized models. The quantized models use lower-precision (e.g. 8-bit instead of 32-bit float), leading to benefits during deployment.

This is achieved by modeling quantization errors during training which helps in maintaining accuracy as compared to FP16 or FP32.

7)what are model parallelism and data parallelism ?why is the latter generally recommended ?

Ans : Model parallelism is a distributed training method in which the deep learning model is partitioned across multiple devices, within or across instances.

Data Parallelism means concurrent execution of the same task on each multiple computing core. Let's take an example, summing the contents of an array of size N. For a single-core system, one thread would simply sum the elements [0] . . . [N − 1].

8)when training a model across multiple servers what distribution strategies can you use ? How do you choose which one to use ?

Ans : Training modern deep learning models requires large amounts of computation, often provided by GPUs. Scaling model training from one GPU to many (so-called distributed training) can enable much faster training and innovation at a lower cost.that you start with a relatively small training dataset that can be trained on a single processor on a single server. As you understand more and more about the business problem you are trying to solve, you realize that you need a more complex model and significantly more data to solve the problem. Consequently, the time needed to train such models on a single GPU single server configuration can grow considerably, sometimes by days or weeks. Distributed training allows you to train large models and models that require large datasets on multiple GPUs and multiple servers in a shorter amount of time and at a much lower computation cost.

Mirrored Strategy-Single Server Multi-GPU: The simplest way to distribute the work is to spread them over multiple GPUs in one single server. Each model is “mirrored” on all GPUs. These variables are kept in sync with each other by applying identical updates. Efficient all-reduce algorithms are used to communicate the variable updates across the devices.

TPU Strategy-Single Server Multi-GPU: “TPU Strategy” allows you to run your TensorFlow training on tensor processing units (TPUs), Google's specialized ASICs designed to accelerate machine learning workloads. In terms of distributed training architecture, TPUStrategy is the same MirroredStrategy-it implements synchronous distributed training. TPUs provide their own implementation of efficient all-reduce and other collective operations across multiple TPU cores, which are used by the TensorFlow API.

MultiWorkerMirroredStrategy-Multi Server

Multi GPU: The next level of distribution happens over multiple servers and multiple GPUs in each server. This strategy still implements the synchronous data parallel approach by replicating all models to each of the available GPUs and uses a multi-worker all-reduce communication method to keep all variables in sync.

Parameter server strategy: This strategy supports parameter servers training on multiple machines. In this setup, some machines are designated as workers and some as parameter servers. Each variable of the model is placed on one parameter server, and computation is replicated across all GPUs of all the workers.