1. What does a SavedModel contain? How do you inspect its content?
2. When should you use TF Serving? What are its main features? What are some tools you can use to deploy it?
3. How do you deploy a model across multiple TF Serving instances?
4. When should you use the gRPC API rather than the REST API to query a model served by TF Serving?
5. What are the different ways TFLite reduces a model’s size to make it run on a mobile or embedded device?
6. What is quantization-aware training, and why would you need it?
7. What are model parallelism and data parallelism? Why is the latter generally recommended?
8. When training a model across multiple servers, what distribution strategies can you use? How do you choose which one to use?

Answer:

1. A SavedModel contains a serialized version of the trained TensorFlow model, along with its variables, assets, and metadata. You can inspect its content using the **saved\_model\_cli** command-line tool, which allows you to inspect the input and output signatures, variables, assets, and other information about the SavedModel.
2. TF Serving is a serving system that allows you to serve TensorFlow models in production. You should use TF Serving when you need to serve large-scale models with low latency, high throughput, and high availability. Its main features include support for multiple models and versions, dynamic model loading, model introspection, and advanced monitoring and logging capabilities. Some tools you can use to deploy TF Serving include Docker, Kubernetes, and TensorFlow Extended (TFX).
3. To deploy a model across multiple TF Serving instances, you can use a technique called model sharding, where you split the model across multiple instances and route the incoming requests to the appropriate instance. You can also use a load balancer to distribute the requests across the instances.
4. The gRPC API is generally faster and more efficient than the REST API, especially for large inputs or outputs, or for high-throughput scenarios. You should use the gRPC API when low latency and high throughput are critical, and when you need to handle large inputs or outputs.
5. TFLite reduces a model's size to make it run on mobile or embedded devices by applying various techniques, such as quantization, pruning, and compression. For example, TFLite can quantize the weights and activations of a model to use 8-bit integers instead of 32-bit floats, which reduces the model size and speeds up the inference.
6. Quantization-aware training is a technique that allows you to train a model with low-precision weights and activations, and then quantize the model during inference to reduce its memory and compute requirements. You would need it if you want to deploy a model on a device with limited resources, such as a mobile or embedded device.
7. Model parallelism and data parallelism are two ways to parallelize the training of large models across multiple devices or machines. Model parallelism splits the model across multiple devices and trains each part separately, while data parallelism trains multiple replicas of the model on different mini-batches of data in parallel. Data parallelism is generally recommended because it is easier to implement and scales better.
8. When training a model across multiple servers, you can use several distribution strategies, such as MirroredStrategy, ParameterServerStrategy, and CentralStorageStrategy. MirroredStrategy trains the model on each device in parallel and synchronizes the variables at each step. ParameterServerStrategy trains the model on a set of worker devices and updates the variables on a set of parameter server devices. CentralStorageStrategy trains the model on a set of worker devices and stores the variables on a central storage device. The choice of distribution strategy depends on the size of the model, the number of devices, and the communication bandwidth.