**DL theory : Assingments-10**

1. A SavedModel contains the model's architecture, variables, and metadata needed for serving or further training. You can inspect its content by using the command-line tool saved\_model\_cli or the Python API tf.saved\_model.load.
2. TF Serving is a system for serving TensorFlow models in a production environment. You should use it when you want to deploy a TensorFlow model in a production environment and take advantage of its features such as auto-scaling, versioning, and monitoring. Some tools you can use to deploy it include TensorFlow Serving, Kubernetes, and Docker.
3. To deploy a model across multiple TF Serving instances, you can use a load balancer to distribute the incoming requests to the different instances. This allows for horizontal scaling and improved performance.
4. The gRPC API should be used when low latency and high throughput are important. It is faster than the REST API and allows for bi-directional streaming.
5. TFLite reduces a model's size to make it run on a mobile or embedded device by using techniques such as weight quantization, pruning, and operator fusion.
6. Quantization-aware training is a technique for training a model that will be quantized for deployment on low-precision devices such as mobile or embedded devices. It involves training the model with a quantization-aware version of the forward pass, which helps to ensure that the quantized model will perform similarly to the original floating-point model.
7. Model parallelism and data parallelism are two strategies for parallelizing the training of a model across multiple devices or servers. Model parallelism involves dividing the model across multiple devices and training each part separately, while data parallelism involves dividing the data across multiple devices and training the model in parallel on each part. Data parallelism is generally recommended because it is more straightforward and easier to implement.
8. When training a model across multiple servers, you can use data parallelism and model parallelism distribution strategies. To choose which one to use, you need to consider the size and complexity of your model and the resources of your servers. If your model is large and complex, you may want to use model parallelism. If your servers have a large amount of memory and computational resources, you may want to use data parallelism.