1. Why would you want to use the Data API?

ANS:

APIs are needed to bring applications together in order to perform a designed function built around sharing data and executing pre-defined processes. They work as the middle man, allowing developers to build new programmatic interactions between the various applications people and businesses use on a daily basis.

1. What are the benefits of splitting a large dataset into multiple files?

ANS:

This reduces storage space and improves performance. It also can allow more flexibility- for example if a table of people had their address stored in a separate table, people could then have an unlimited number of addresses- something that could not be accomplished if one tried to store the addresses in the same table.

1. During training, how can you tell that your input pipeline is the bottleneck? What can you do to fix it?

ANS:

If your pipeline is the bottleneck, what can you do to fix it? You can fix by making sure it reads and preprocesses the data in multiple threads in parallel, and prefetches a few batches.

1. Can you save any binary data to a TFRecord file, or only serialized protocol buffers?

ANS:

The TFRecord format is a simple format for storing a sequence of binary records. Protocol buffers are a cross-platform, cross-language library for efficient serialization of structured data. Protocol messages are defined by . proto files, these are often the easiest way to understand a message type.

Most often we have labeled data in PASCAL VOC XML or COCO JSON. Creating a TFRecord file from this data requires following a multistep process: (1) creating a TensorFlow Object Detection CSV (2) Using that TensorFlow Object Detection CSV to create TFRecord files

1. Why would you go through the hassle of converting all your data to the Example protobuf format? Why not use your own protobuf definition?

ANS:

Protocol Buffers (Protobuf) is a free and open-source cross-platform data format used to serialize structured data. It is useful in developing programs to communicate with each other over a network or for storing data.

In protobuf, the payload is smaller, plus the math is simple, and the member-lookup is an integer (so: suitable for a very fast switch /jump)

1. When using TFRecords, when would you want to activate compression? Why not do it systematically?

ANS:

The TFRecord format is a simple format for storing a sequence of binary records. Protocol buffers are a cross-platform, cross-language library for efficient serialization of structured data. Protocol messages are defined by . proto files, these are often the easiest way to understand a message type.

The rule of thumb is to have at least 10 times as many files as there will be hosts reading data. At the same time, each file should be large enough (at least 10+MB and ideally 100MB+) so that you benefit from I/O prefetching. For example, say you have X GBs of data and you plan to train on up to N hosts.

In order to use a Dataset we need three steps:

1. Importing Data. Create a Dataset instance from some data.
2. Create an Iterator. By using the created dataset to make an Iterator instance to iterate through the dataset.
3. Consuming Data. By using the created iterator we can get the elements from the dataset to feed the model.
4. Data can be preprocessed directly when writing the data files, or within the tf.data pipeline, or in preprocessing layers within your model, or using TF Transform. Can you list a few pros and cons of each option?

ANS:

Preprocessing data with TensorFlow Transform

1. Create an input function for training.
2. Train, Evaluate the model.
3. Transform new data.
4. Export the model.

The tf. data API enables you to build complex input pipelines from simple, reusable pieces. For example, the pipeline for an image model might aggregate data from files in a distributed file system, apply random perturbations to each image, and merge randomly selected images into a batch for training.

When performing regression or classification, which of the following is the correct way to preprocess the data? Explanation: You need to always normalize the data first. If not, PCA or other techniques that are used to reduce dimensions will give different results.