1. How would you describe TensorFlow in a short sentence? What are its main features? Can you name other popular Deep Learning libraries?

ANS:

TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data. TensorFlow works on the basis of data flow graphs that have nodes and edges

Some of the key features of TensorFlow are: Efficiently works with mathematical expressions involving multi-dimensional arrays. Good support of deep neural networks and machine learning concepts. GPU/CPU computing where the same code can be executed on both architectures.

1. Is TensorFlow a drop-in replacement for NumPy? What are the main differences between the two?

ANS:

Tensorflow is a library for artificial intelligence, especially machine learning. Numpy is a library for doing numerical calculations.

1. Do you get the same result with tf.range(10) and tf.constant(np.arange(10))?

ANS:

In TensorFlow the differences between constants and variables are that when you declare some constant, its value can't be changed in the future (also the initialization should be with a value, not with operation). Nevertheless, when you declare a Variable, you can change its value in the future with tf.

tf.range(limit, delta=1, dtype=None, name='range')  
tf.range(start, limit, delta=1, dtype=None, name='range')

Creates a sequence of numbers that begins at start and extends by increments of delta up to but not including limit .

1. Can you name six other data structures available in TensorFlow, beyond regular tensors?

ANS:

Tensors are the basic data structures in TensorFlow, and they represent the connecting edges in a dataflow graph. A tensor simply identifies a multidimensional array or list. The tensor structure can be identified with three parameters: rank, shape, and type. Rank: Identifies the number of dimensions of the tensor.

1. A custom loss function can be defined by writing a function or by subclassing the keras.losses.Loss class. When would you use each option?

ANS:

A custom loss function can be created by defining a function that takes the true values and predicted values as required parameters. The function should return an array of losses. The function can then be passed at the compile stage.

For a loss function, we need the model's actual value and the predicted value to compare and calculate the loss value. In Tensorflow, we will write a custom loss function that will take the actual value and the predicted value as input. This custom loss function will subclass the base class “loss” of Keras.

The mean squared error loss function can be used in Keras by specifying 'mse' or 'mean\_squared\_error' as the loss function when compiling the model. It is recommended that the output layer has one node for the target variable and the linear activation function is used.

1. Similarly, a custom metric can be defined in a function or a subclass of keras.metrics.Metric. When would you use each option?

ANS:

Below is a list of the metrics that you can use in Keras on regression problems.

* Mean Squared Error: mean\_squared\_error, MSE or mse.
* Mean Absolute Error: mean\_absolute\_error, MAE, mae.
* Mean Absolute Percentage Error: mean\_absolute\_percentage\_error, MAPE, mape.
* Cosine Proximity: cosine\_proximity, cosine.

The loss function is used to optimize your model. This is the function that will get minimized by the optimizer. A metric is used to judge the performance of your model. This is only for you to look at and has nothing to do with the optimization process.

1. When should you create a custom layer versus a custom model?

ANS:

1. On this page.
2. Setup.
3. The Layer class: the combination of state (weights) and some computation.
4. Layers can have non-trainable weights.
5. Best practice: deferring weight creation until the shape of the inputs is known.
6. Layers are recursively composable.
7. The add\_loss() method.
8. The add\_metric() method.

Keras - Customized Layer

1. Step 1: Import the necessary module. First, let us import the necessary modules − from keras import backend as K from keras. ...
2. Step 2: Define a layer class. ...
3. Step 3: Initialize the layer class. ...
4. Step 4: Implement build method. ...
5. Step 5: Implement call method. ...
6. Step 6: Implement compute\_output\_shape method.
7. What are some use cases that require writing your own custom training loop?

ANS:

Here's our training loop:

1. We open a for loop that iterates over epochs.
2. For each epoch, we open a for loop that iterates over the dataset, in batches.
3. For each batch, we open a GradientTape() scope.
4. Inside this scope, we call the model (forward pass) and compute the loss.

When you need to customize what fit() does, you should override the training step function of the Model class. This is the function that is called by fit() for every batch of data.

1. Can custom Keras components contain arbitrary Python code, or must they be convertible to TF Functions?

ANS:

When you need to customize what fit() does, you should override the training step function of the Model class. This is the function that is called by fit() for every batch of data. You will then be able to call fit() as usual -- and it will be running your own learning algorithm.

The Functional API: The Functional API is the most popular method to build Keras models. It can do everything that the Sequential API can do. Also, it allows multiple inputs, multiple outputs, branching, and layer sharing.

1. What are the main rules to respect if you want a function to be convertible to a TF Function?

ANS:

Better performance with tf. function

* Executing Python side effects.
* All outputs of a tf.function must be return values.
* Recursive tf.functions are not supported.

TensorFlow implements standard mathematical operations on tensors, as well as many operations specialized for machine learning. Note: Typically, anywhere a TensorFlow function expects a Tensor as input, the function will also accept anything that can be converted to a Tensor using tf

1. When would you need to create a dynamic Keras model? How do you do that? Why not make all your models dynamic?

ANS:

Keras provides three different ways to build deep learning models - Sequential, Functional API, and Model subclassing. Each of the techniques addresses different use cases.

Regularizers: It will help you to optimize the model and the layer. It dynamically applies penalties on the weight. It works during the optimization process.