Proteinea Report

Q1 Answers:

A) We need 32 gpus to handle a 32 batch_size

B) To achieve this type of training on a cluster of GPUs we will need to use the ***tf.distribute*** module provided by ***Tensorflow.***
The ***tf.distribute*** module provides different types of training ***strategies*** for different use cases. In our case we need to use a cluster of GPUs as a shared memory to handle the 32 batch_sze, this scenario is called **data parallelism.**
So the strategy that fits our needs is ***tf.distribute.MirroredStrategy:*** will use it as follow:
   - Instantiate a MirroredStrategy, optionally configuring which specific devices you want to use (by default the strategy will use all GPUs available).
   - Use the strategy object to open a scope, and within this scope, create all the Keras objects you need that contain variables. Typically, that means creating & compiling the model inside the distribution scope.
   - Train the model via fit() as usual.
   - Resource: keras documentation

C) I will use the ***nvidia-smi*** cli utility to show if all my gpus are working fine. If it shows only a single **gpu,** then the issue will be **related to drivers**. So after that I will check my driver installation and make sure all my devices appear when I run the above command again.

Q2 Answers:

A) There is lots of randomization goes in deep learning ranging from (randomly splitting the data, randomly initializing the weights, ..etc) but here let's only consider our example:
Usually there is a randomization in picking up our data set but in the provided keras example the load_data() method fortunately doesn't do that.
Skimming the code reaching the model building step there isn't anything that produces different results. So let's jump directly into the model building: the conv2d layers consists of a set of 3X3 kernels think of them as matrices those matrices have initial values, and to your surprise these values get initialized using the default initialization method **glorot uniform** which produces ***random values*** (random values leads to random results)***.***
Also the example uses ***adam*** as an optimizer algorithm, and ***adam*** is a stochastic algorithm (This means that its behavior incorporates elements of randomness.) The impact is that each time the stochastic machine learning algorithm is run on the same data, it learns a slightly different model and that is a feature not a bug, as it's useful when working with hard problems.