Assignment-2 Solutions

Question 1 :

In PyTorch , we calculate gradients using a dynamic computational graph.

- Behaviour of loss.backward() : when we call this method,Pytorch computes the gradient of the loss with respect to all leaf tensors that have reuires_grad=True.These values are stored in the .grad which is an attribute of each tensor.
- Pytorch adds the newly calculated gradients to the existing values in the .grad attribute rather than over writing them.
- The requirement of optimizer.zero_grad() : Because the gradients accumulate , if we donot clear them at the start of every training step,the gradients from the previous batch will be added to the current ones.

  If forgotten the update rule becomes

  $\theta_{new} = \theta_{old} - \eta \cdot \sum_{i=1}^{current\_step} \nabla Loss_i$

The above formula causes thr gradients to grow incorrectly , leading to divergent training or incorrect weight updates.

Question 2:

.view():

It requires sthe tensor to be contiguous in memory. It always returns a “view” of the original data(no copying).It will give an error if the tensor is non contiguous like after a transpose

.reshape()

It can handle both contiguous and non contiguous tensors. It returns a view if possible , otherwise it copies the data to a new memory block.We will have safer fall back because it handles the memory layout automatically.

Question-3:

If we attempt an operation(like addition) between a tensor on the CPU and a tensor on CUDA:0,Pytorch will then raise a runtime error.All tensors involved in a single operation must reside on the same device.

Tensor Creation in forward : Moving a model to the GPU via model.to('cuda') only moves the model's existing parametes and buffers. Any new tensors created inside the forward method like example torch.zeros(5) will be created on the CPU by default.You must manually assign them to the correct device using device=x.device or .to(device).