Accumulated Gradients: Iterative Learning Across Epochs

While `optimizer.zero_grad()` resets the gradients to zero at the beginning of each epoch, it doesn't mean that each epoch is completely independent and disconnected from the previous epochs.

The learning process is iterative, and the model parameters (weights and biases) are updated based on the accumulated gradients from all the training data seen during multiple epochs. Let me explain the learning process step by step:

1. **Initialization**: At the beginning, the model's parameters (weights `W` and biases) are initialized with random values.
2. **Epochs**: The training data is divided into epochs, and for each epoch, the model goes through the entire training dataset.
3. **Gradient Accumulation**: During each epoch, the gradients for the model parameters are computed for each batch of data, using the current batch's data. The gradients are accumulated over all batches within the epoch.
4. **Parameter Update**: After the completion of each epoch, the optimizer uses the accumulated gradients to update the model parameters (weights and biases). The update rule typically involves subtracting a scaled value of the gradients from the current parameter values.
5. **Next Epoch**: At the start of the next epoch, the gradients are reset to zero using `optimizer.zero_grad()`, and the process is repeated for the new epoch. However, the model's parameters are not reset to their initial random values; instead, they retain the updated values from the previous epoch.
6. **Iterative Learning**: Over multiple epochs, the model learns from the entire dataset, and the updates from each epoch gradually refine the model's parameters to better fit the training data.

In this way, each epoch contributes to the learning process, and the model gradually improves its performance as it sees more data and learns from the accumulated gradients over multiple epochs. The learning process is iterative and builds on the knowledge gained from previous epochs to refine the model's parameters and make better predictions on new data.

In summary, while the gradients are reset at the beginning of each epoch to avoid interference between epochs, the learning process is cumulative, and the model builds knowledge from all the data seen during multiple epochs.

ABOUT BATCHES IN EACH EPOCH --

After all batches have been processed within the epoch, the accumulated gradients from all batches are used to update the model's parameters in a parameter update step. This parameter update step typically involves taking a weighted average of the gradients from all batches to update the model's parameters.

Regarding your question about making gradients zero for each batch: No, you do not need to manually make gradients zero for each batch. PyTorch takes care of it automatically. When you call `optimizer.zero_grad()` at the beginning of each epoch, it sets all gradients of the model's parameters to zero. After the completion of each iteration (batch) within the epoch, the gradients for the current batch are computed and accumulated, but gradients from previous batches within the same epoch are not retained.

So, you don't need to worry about making gradients zero for each batch separately; `optimizer.zero_grad()` takes care of that at the beginning of each epoch, ensuring that each batch starts with fresh gradients.