1. **Stochastic Gradient Descent (SGD)** for each training example or (x,y) pair:

   - Suppose we have 2 hidden layers. For each training example, we perform a forward pass, calculate the loss, and backpropagate to compute the gradients for the weights, such as:

   $$\frac{\partial \text{Loss}}{\partial W_1}, \frac{\partial \text{Loss}}{\partial W_2}, \frac{\partial \text{Loss}}{\partial W_3}$$

   - We then update the weights immediately after each example:

   $$W_1 = W_1 - \eta \frac{\partial \text{Loss}}{\partial W_1}, \quad W_2 = W_2 - \eta \frac{\partial \text{Loss}}{\partial W_2}, \quad W_3 = W_3 - \eta \frac{\partial \text{Loss}}{\partial W_3}$$

   - This method updates frequently, but it could introduce instability and may lead to overfitting due to the continuous updating.

2. **SGD but Batch Learning**:

   - In contrast, batch learning waits until the entire batch (or minibatching) is processed. After the forward pass, we compute the average gradients for each weight over the entire batch. For example, the first layer would have the average gradient loss with respect to $W_1$:

   $$\frac{1}{N} \sum_{i=1}^{N} \frac{\partial \text{Loss}_i}{\partial W_1}$$

   - After averaging the gradients, we update the weights, for example, for hidden layer 1 and weight 1 as follows:

   $$W_1 = W_1 - \eta \cdot \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \text{Loss}_i}{\partial W_1}$$

   This requires waiting for the entire batch (or minibatches) to be processed before making the updates.

Which one is more reasonable? Why?