

# 2Background

Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Block-wise k-bit Quantization Quantization is the process of discretizing an input from a rep- resentation that holds more information to a representation with less information. It often means taking a data type with more bits and converting it to fewer bits, for example from 32-bit floats to 8-bit Integers. To ensure that the entire range of the low-bit data type is used, the input data type is commonly rescaled into the target data type range through normalization by the absolute maximum of the input elements, which are usually structured as a tensor. For example, quantizing a 32-bit Floating Point (FP32) tensor into a Int8 tensor with range [—127, 127]:

127 XInt8 = round ( round(CFP32 . xFP32), (absmax(XFP32)

(1)

where c is the quantization constant or quantization scale. Dequantization is the inverse: XInt8 =XFP32 CFP32

(2)

The problem with this approach is that if a large magnitude value (i.e., an outlier) occurs in the input tensor, then the quantization bins—certain bit combinations- -are not utilized well with few or no numbers quantized in some bins. To prevent the outlier issue, a common approach is to chunk the input tensor into blocks that are independently quantized, each with their own quantization constant c. This can be formalized as follows: We chunk the input tensor X E Rbxh into n contiguous blocks of size B by flattening the input tensor and slicing the linear segment into n = (b x h)/ B blocks. We quantize these blocks independently with Equation 1 to create a quantized tensor and n quantization constants Ci. Low-rank Adapters Low-rank Adapter (LoRA) finetuning [28] is a method that reduces memory requirements by using a small set of trainable parameters, often termed adapters, while not updating the full model parameters which remain fixed. Gradients during stochastic gradient descent are passed through the fixed pretrained model weights to the adapter, which is updated to optimize the loss function. LoRA augments a linear projection through an additional factorized projection. Given a projection XW = Y with X E Rbxh, W E Rhxo LoRA computes:

where L1 E Rhxr and L2 E Rrx°, and s is a scalar.

Y = XW + sXLL2

(3)

Memory Requirement of Parameter-Efficient Finetuning One important point of discussion is the memory requirement of LoRA during training both in terms of the number and size of adapters used. Since the memory footprint of LoRA is so minimal, we can use more adapters to improve performance without significantly increasing the total memory used. While LoRA was designed as a

3